TECHNISCHE
UNIVERSITÄT
DARMSTADT

DECENTRALIZED MONITORING *in* MOBILE AD HOC NETWORKS –

Provisioning of Accurate and Location-Aware Monitoring Information

Vom Fachbereich Elektrotechnik und Informationstechnik
der Technischen Universität Darmstadt
zur Erlangung des akademischen Grades eines
Doktor-Ingenieurs (Dr.-Ing.)
genehmigte Dissertation

von

DIPL.-INFORM. DOMINIK STINGL

Geboren am 5. Oktober 1981 in Mainz

Vorsitz: Prof. Dr.-Ing. Helmut Schlaak
Referent: Prof. Dr.-Ing. Ralf Steinmetz
Korreferent: Prof. Dr.-Ing. Michael Zink

Tag der Einreichung: 18. Juli 2014
Tag der Disputation: 16. Dezember 2014

Hochschulkennziffer D17
Darmstadt 2015

ABSTRACT

There has been an immense popularity increase of mobile communication devices, such as smartphones and tablet PCs, due to the technical progress in equipment manufacturing and communication. Paired with the constantly growing availability of wireless broadband access over cellular networks the devices enable the continuous consumption of applications. Furthermore, they foster new types of applications, such as location-based services or mobile social networks, where the user interacts with its vicinity or nearby users. Due to the increasing popularity paired with the continuous consumption of applications cellular networks can hardly handle the resulting traffic that frequently exceeds their capacity. To unburden these networks *mobile ad hoc networks* represent a viable alternative, complementing or superseding cellular networks. Exploiting the growing density of mobile communication devices, mobile ad hoc networks rely on the direct interconnection of devices and provide an useful substrate to exchange information and to deploy applications. In particular, the direct device interaction reflects the communication and interaction pattern of location-based services and mobile social networks, which we term as *locality of interaction*.

Despite the fact that mobile ad hoc networks represent a viable communication substrate, they are exhibiting highly dynamic characteristics, which are mainly attributable to the autonomous behavior of users. To handle the dynamic nature mobile ad hoc networks must be adapted to the current state of the network and the influencing conditions of the surrounding environment. In this regard, *monitoring* constitutes an inevitable aspect of mobile ad hoc networks. It collects information from the users and provides essential insights into the current network state, serving as basis to adapt the network. As mobile ad hoc networks are exclusively established by users without a central entity the users themselves must monitor the network and exploit the obtained monitoring information to adapt the network. Consequently, the issues arise that the users are in charge to measure and collect the monitoring information as well as to distribute the obtained insights among them. In addition to the information about the network state a user requires detailed information about its vicinity. The reason for the provisioning of location-dependent information results from the locality of interaction in mobile ad hoc networks: as a user can only interact with its neighborhood it particularly requires detailed monitoring information about its vicinity. In the context of monitoring we refer to this property as *location-aware monitoring*.

To address the presented issues *decentralized monitoring mechanisms* have been introduced and developed. The participating users accomplish the mentioned tasks of monitoring and provide the required monitoring information. However, existing approaches for decentralized monitoring in mobile ad hoc networks exhibit certain shortcomings or make some limiting assumptions. Examples comprise (i) the need for a decentralized infrastructure with dedicated devices that monitor the network or (ii) the expectation of an additional external entity (e.g., a network operator, which further processes the monitoring information). The examples reveal that existing

approaches do not comply with the addressed issues and are not suited for the deployment in the envisaged scenarios. Consequently, the major objective of this thesis is the design of decentralized monitoring approaches that tackle these issues and are applicable in the corresponding scenarios.

BlockTree.KOM represents our first solution for decentralized monitoring in mobile ad hoc networks. It proposes a set of four fundamental concepts that specify its underlying structure and the exchange of information over that structure. In detail, BlockTree.KOM relies on a hierarchical topology and on its own tailored communication methods to monitor the communication network. With P-BlockTree.KOM and C-BlockTree.KOM we present two different approaches of BlockTree.KOM that comply with the presented concepts but implement them in different ways. Based on an extensive evaluation, the obtained results reveal that both approaches provide accurate monitoring information despite the dynamic nature of mobile ad hoc networks. Exploiting the hierarchical topology, specifically P-BlockTree.KOM is able to provide location-aware monitoring information: every user obtains an accurate view of its vicinity as well as summarized insights on distant regions.

Mobi-G.KOM constitutes our second solution for decentralized monitoring in mobile ad hoc networks. Its design fundamentally differs from BlockTree.KOM, since it operates on a flat topology and abstains from the establishment of a hierarchy. To implement the communication between users over the flat topology Mobi-G.KOM exploits the robust communication pattern of gossiping. With the flat design of Mobi-G.KOM we sacrifice the provisioning of location-aware monitoring information. The obtained results reveal that the view of a user on nearby or distant regions does not differ and leads to inaccurate information about a user's vicinity. However, the experiments unveil the advantages of the flat design paired with the robust communication pattern of gossiping. Mobi-G.KOM is highly robust and operates in highly dynamic environments. Furthermore, the provisioning of accurate monitoring information comes at considerably lower cost: Mobi-G.KOM neither produces traffic to maintain a hierarchical topology nor transmits data at multiple levels of the topology.

KURZFASSUNG

Mobile Kommunikationsgeräte, wie Smartphones oder Tablets, haben in den vergangenen Jahren stark an Popularität gewonnen. Dies ist zum einen dem technologischen Fortschritt der Geräte und zum anderen der wachsenden Verfügbarkeit mobiler Breitbandzugänge über zelluläre Infrastrukturen zuzurechnen. Dank dieser Tatsachen können die Nutzer von überall auf ihre Daten zugreifen. Zusätzlich ermöglichen die mobilen Kommunikationsgeräte die Verwendung standortbezogener Dienste oder mobiler sozialer Applikationen, die sich primär durch eine verstärkte Interaktion mit der unmittelbaren Umgebung und den dazugehörigen Inhalten auszeichnen. Durch die steigende Anzahl mobiler Kommunikationsgeräte sowie deren intensiver Nutzung werden jedoch die zugrundeliegenden zellulären Infrastrukturen häufig über ihre Leistungsgrenzen hinweg belastet. In diesem Zusammenhang bieten sich *mobile Ad-Hoc-Netze* als ideale Ergänzung oder Alternative an, um zelluläre Infrastrukturen von der resultierenden Datenflut zu entlasten. Mobile Ad-Hoc-Netze basieren auf der direkten Vernetzung der mobilen Kommunikationsgeräte und bieten gerade in dicht besiedelten Gebieten eine funktionsfähige Grundlage zum Informationsaustausch. Die direkte Interaktion zwischen den Kommunikationsgeräten reflektiert insbesondere die Interaktionsmuster standortbezogener Dienste und mobiler sozialer Applikationen, welche wir als *lokale Interaktion* bezeichnen.

Obwohl mobile Ad-Hoc-Netze eine sinnvolle Kommunikationsgrundlage darstellen, haben sie mit einer hohen Dynamik zu kämpfen, die maßgeblich aus der Vernetzung mobiler Kommunikationsgeräte autonomer Nutzer resultiert. Um der hohen Dynamik Herr zu werden, müssen sich mobile Ad-Hoc-Netze dem sich stetig ändernden Netzzustand und den Einflüssen aus der Umgebung anpassen. In diesem Zusammenhang stellt *Monitoring* eine Kernkomponente mobiler Ad-Hoc-Netze dar, das auf Basis gesammelter Informationen Einblicke in den aktuellen Zustand des Netzes ermöglicht und als Grundlage für mögliche Adaptionen dient. Aufgrund der Tatsache, dass mobile Ad-Hoc-Netze ausschließlich durch autonome Nutzer administriert werden, muss die Aufgabe des Monitorings ebenfalls durch die Nutzer erbracht werden. Man spricht in diesem Fall auch von *dezentralem Monitoring*. Im Rahmen des dezentralen Monitorings müssen die Nutzer selbst wichtige Daten über den Zustand des Netzes messen, sammeln und anschließend als Informationsgrundlage im Netz zur Verfügung stellen. Neben der Erfassung des Zustands des Kommunikationsnetzes ist es ebenso wichtig, dass ein Nutzer detaillierte Informationen über seine direkte Umgebung bekommt. Der Grund dafür resultiert aus der vorher erwähnten lokalen Interaktion, da ein Nutzer in jedem Fall auf seine direkte Nachbarschaft angewiesen ist, um zu kommunizieren und zu interagieren. Im Falle von Monitoring bezeichnen wir diese Eigenschaft auch als *standortbezogenes Monitoring*.

Es existieren mehrere Ansätze für dezentrales Monitoring, welche die oben beschriebenen Anforderungen und Aufgaben umsetzen. Jedoch weisen diese gewisse Einschränkungen auf oder treffen nicht umsetzbare Annahmen: Beispielsweise gibt es Ansätze, die eine dezentrale Monitoring-Architektur mit dedizierten Geräten zum

Messen des Netzwerkzustands voraussetzen oder eine dritte Partei in Form eines Netzwerkadministrators zur Datenauswertung vorsehen. Anhand dieser Beispiele wird deutlich, dass diese Ansätze die beschriebenen Anforderungen und Aufgaben nicht oder nur teilweise umsetzen und daher in den von uns betrachteten Szenarien nicht einsetzbar sind. Dementsprechend fokussiert diese Arbeit auf die Entwicklung passender Ansätze für dezentrales Monitoring in mobilen Ad-Hoc-Netzen, die den erwähnten Anforderungen und betrachteten Szenarien genügen.

Mit BlockTree.KOM präsentieren wir unseren ersten Lösungsvorschlag für dezentrales Monitoring in mobilen Ad-Hoc-Netzen. BlockTree.KOM basiert auf vier grundlegenden Konzepten, welche die zugrundeliegende Struktur sowie den darüber erfolgenden Informationsaustausch beschreiben. Bei der Struktur von BlockTree.KOM handelt es sich um eine hierarchische Topologie, während für den Informationsaustausch eigene, maßgeschneiderte Kommunikationsverfahren umgesetzt werden. P-BlockTree.KOM und C-BlockTree.KOM stellen zwei konkrete Realisierungen der Konzepte von BlockTree.KOM dar, die sich aber in ihrer Umsetzung voneinander unterscheiden. Die ausführliche Evaluation zeigt, dass beide Ansätze akkurate Informationen über den Zustand des Kommunikationsnetzes trotz der vorherrschenden Dynamik liefern. Dank der zugrundeliegenden hierarchischen Topologie ermöglicht vor allem P-BlockTree.KOM standortbezogenes Monitoring und liefert einen granularen Überblick über den aktuellen Zustand der unmittelbaren Nachbarschaft eines Nutzers, während aggregierte Informationen über den aktuellen Netzzustand entfernter Regionen vorliegen.

Mobi-G.KOM beschreibt unseren zweiten Ansatz für dezentrales Monitoring in mobilen Ad-Hoc-Netzen, welcher sich jedoch grundlegend von BlockTree.KOM unterscheidet, da er beispielsweise eine flache anstelle einer hierarchischen Topologie verwendet. Zum Informationsaustausch zwischen den Nutzern greift Mobi-G.KOM auf Gossiping zurück, welches ein Konzept für robuste Kommunikation darstellt. Aufgrund der flachen Topologie ist es Mobi-G.KOM nicht möglich standortbezogenes Monitoring zu realisieren und granulare Ergebnisse über die nähere Umgebung eines Nutzers zu liefern. Die erhaltenen Ergebnisse zeigen, dass es keinen Unterschied zwischen den zur Verfügung stehenden Informationen über die nähere Umgebung sowie entfernte Regionen gibt, wodurch vor allem der ermittelte Zustand der näheren Umgebung von dem tatsächlichen abweicht. Auf der anderen Seite unterstreichen die durchgeführten Experimente die Vorteile der flachen Topologie in Kombination mit dem robusten Informationsaustauschs durch Gossiping. Mobi-G.KOM stellt einen äußerst robusten Ansatz dar, der in hochdynamischen Szenarien akkurate Ergebnisse liefert. Zusätzlich ist der resultierende Aufwand zur Erfassung des Zustands mobiler Ad-Hoc-Netze deutlicher geringer, da die Aufrechterhaltung einer flachen Topologie keinen Aufwand verursacht und die Daten nur über ein Level der flachen Hierarchie übertragen werden.

CONTENTS

1 INTRODUCTION

The technological progress in the area of mobile communication has led to a triumph of mobile communication devices in our daily life. Recent statistics [36] reveal that 1.9 billion smartphones have been sold worldwide until the end of 2013, exceeding two billion mobile broadband subscriptions. Based on these figures, it is predicted that the number of smartphones will surpass 5.6 billion by 2019 with more than eight billion mobile broadband subscriptions. To a large extent the overwhelming success results from the versatile capabilities of those devices. Equipped with a powerful CPU, large storage, and diverse sensors, they enable the utilization of services and applications that have been originally designed for desktop PCs or notebooks. Paired with the growing availability of wireless broadband access over cellular networks [36] users may consume these services and applications anytime and anywhere. Consequently, sale statistics reveal that the number of sold mobile communication devices already superseded desktop PCs and notebooks in 2011 [17].

These mobile communication devices facilitate the consumption of well-known applications and services such as web browsing or real-time entertainment, which account for the majority of the global mobile data traffic [147]. Furthermore, the devices boost new types of applications and services, comprising so-called location-based services [178, 153, 74, 32] or mobile social networking applications [97, 134, 141, 63]. Location-based services exploit the current location of the user and permit a user to consume and generate information about its vicinity. A recent study [186] in the U.S. on the utilization of smartphones has shown that 74 % of smartphone users (up from 55 % in 2011) use their device to obtain information based on their current location. In mobile social networking applications socially related users come together to exchange information. The popularity of these applications is reflected by the resulting traffic [147]: despite the usually small size of the exchanged information, mobile social networking applications rank among the top-three drivers for mobile data traffic in North America, Europe, and Latin America.

As estimated by the Cisco Visual Networking Index [25] the intense utilization of mobile communication devices resulted in a global mobile data traffic of 1.5 Exabyte per month until the end of 2013. The study predicts that the traffic will increase by a factor of 11 and reach 15.9 Exabyte by 2018. The resulting load of the mobile communication devices must be carried by the cellular networks that provide the infrastructure for the wireless broadband access. As witnessed in the past [73, 177, 98, 168] cellular networks suffer from the high and data-intensive utilization. Particularly in urban areas with frequented spots or during crowded events, cellular networks are overloaded and exhibit a degrading performance for the transmission of voice and data [150]. Exploiting the density of mobile communication devices, a so-called *mobile ad hoc network (MANET)* constitutes a viable alternative in addition to or as a substitute for the cellular infrastructure. Based on the direct communication between the devices, either using Bluetooth or the ad hoc mode from the *wireless local area network (WLAN)* interface, MANETs provide an useful substrate to exchange information.

Similar to *peer-to-peer (P2P)* networks [125, 154] there is no infrastructure or central entity that manages the network and the deployed application or organizes the communication between the participating users. Instead, following the P2P paradigm every participating user must offer and provide its resources for the maintenance of the MANET to benefit from the deployed application or service. Typical examples for the deployment in MANETs range from data-intensive applications [54, 80, 6] to mobile social networks [97, 134] or location-based services [19, 126]. In this regard, data-intensive applications, such as real-time entertainment, profit from offloading a cellular network [54, 6, 11]. For mobile social networks and location-based services [63] the direct interaction between the devices reflects the local communication and interaction pattern of these applications. For the remainder of this thesis we specifically focus on location-based services, which serve as an application example for the thesis and as data basis for the evaluation.

1.1 MOTIVATION

Though MANETs provide an useful communication substrate and alternative to infrastructure-based networks, they suffer from their highly dynamic nature. Similar to P2P networks MANETs rely on the direct interconnection of users and must handle the autonomous behavior, which is reflected by frequent arrivals and departures of users. In addition, users are mobile and not bound to a location but move around, leading to frequently changing conditions of the environment, which a MANET must face. Examples comprise

- places or spots with an increasing or decreasing user density;
- the spatial expansion or contraction of the network;
- the appearance of fast or slow moving users.

Paired with the limited communication range interconnections between users may not last long, hence, leading to a constantly changing network topology with permanently varying neighbors to communicate. Thus, MANETs must be adapted to the current and constantly changing state of the network and to the external influences arising from the environment. To implement this adaptation it is mandatory to (i) determine the current state and (ii) distribute this information among the participating users so that they may react accordingly. In this regard *monitoring* constitutes an inevitable element of MANETs because it enables the determination and provisioning of the current state of the network and its participating users. For instance, monitoring a MANET helps to capture the access frequency of requested content to identify popular content and to adjust the replication of content accordingly [99]. Furthermore, based on monitoring, the user density in the network may be ascertained to replace the currently deployed routing protocol with an appropriate one [144].

Several approaches have already been designed to monitor MANETs in order to collect information about the network and its participating users [21, 105, 119, 97, 50]. The approaches have in common that they refrain from a central entity or authority, which organizes and manages the process of monitoring. Instead, the approaches implement *decentralized monitoring* and distribute the process of monitoring over the participating users in the network, which are in charge to collect and process the data but also to disseminate the monitored information.

A large fraction of approaches for decentralized monitoring establishes a hierarchy, which is used to collect the monitored information. The application of these approaches in the presented application scenarios, as sketched above, generates two issues.

- A set of the hierarchical approaches [137, 79, 142] requires an already existing infrastructure, consisting of dedicated devices to create and maintain the hierarchy.
- The majority of hierarchical approaches is mainly concerned with the efficient collection of data. The users only capture relevant data and participate to collect the data. In contrast, the design of efficient information dissemination to the remaining users is of little importance.
  - Often, existing approaches assume an additional external instance or entity, e.g., a network operator, which analyzes the collected data to gain insights into the current state and situation [137, 142].
  - Other approaches [21, 97, 8] rely on a set of selected users, which collect and process the data. Consequently, information about the current network state is only available at this set of users, which must serve the remaining users by responding to raised requests. Furthermore, information is only available if the set of selected users is reachable.

Complementary to hierarchical approaches flat approaches for MANETs have been introduced, which rely on the robust communication pattern of gossiping for the concurrent dissemination of information during the collection of data. In contrast to the variety of hierarchical solutions only a small number of flat monitoring approaches has been developed for mobile networks. These approaches do not directly address MANETs but focus on static wireless networks [41], on delay tolerant networks [50], or make simplified assumptions regarding the scenario [180].

The brief overview on existing approaches for monitoring in MANETs reveals several shortcomings for the deployment in the envisaged scenarios. Whereas former approaches mainly tried to address a decentralized collection of monitored data with a possible transfer to external entities, monitoring information is required by the participating users in the network. The information is utilized to obtain a recent view of the current state of the communication network for potential adaptations.

1.2 RESEARCH CHALLENGES

Based on the introduction of decentralized monitoring in MANETs, the following five research challenges are introduced, serving as guideline during the development and evaluation of our targeted approaches for decentralized monitoring. In the following we identify the causes of these challenges, formulate them, and sketch the resulting implications on decentralized monitoring.

Challenge 1: *The decentralized nature of MANETs.*

MANETs are built on top of a group of users' communication devices that contribute their resources to establish and maintain the network as communication substrate.

There is no central instance, which coordinates and manages the process. As a result a monitoring approach does not rely on a central instance that serves as focal point for the collection, processing, and dissemination of monitoring data. Instead, existing approaches decentralize the process of monitoring and operate without the central instance to organize the participating users for two tasks: (i) decentralized capturing and processing of the monitored data, which are spread over the whole network, and (ii) provisioning of the current state to the users in the network.

Challenge 2: *The dynamic nature of MANETs.*

MANETs consist of autonomous and mobile users and are subject to constant changes in several aspects. The autonomous behavior results in a continuous arrival and departure of users, which influences the spatial size of the network or its user density. The inherent mobility amplifies these effects and additionally influences the interconnection between users, leading to a permanently changing network topology. Consequently, decentralized monitoring must withstand the dynamic nature of these networks and operate in networks of different spatial sizes, collect and disseminate data in sparsely or densely populated networks, and handle the constantly changing topology.

Challenge 3: *The wireless communication medium.*

MANETs rely on the direct interaction between users over a wireless communication medium, using for instance Bluetooth or the ad hoc mode of the WLAN interface from the communication device. As a result the device just communicates with neighboring users, which reside inside the communication range. Despite the limited communication range restricted to neighboring users, larger parts of the network must be monitored to provide a meaningful view of the current network state based on the collected data from many users. Furthermore, the exchange of information takes place over an error-prone communication medium with limited transport capacity. Hence, the successful transmission of data must be ensured without the excessive allocation of the communication medium.

Challenge 4: *The limited energy resources.*

Due to the mobility the considered communication devices are not line but battery powered. The limited amount of energy of a device constitutes another scarce resource. In addition to the energy consumption that arises from the actual data transmission monitoring burdens this scarce resource. Consequently, the arising monitoring traffic must be reduced to the transmission of relevant information to preserve the energy resources of the devices in addition to the limited transport capacity of the wireless communication medium.

Challenge 5: *The distance-aware relevance of information.*

The vicinity is of high relevance for a user in a MANET, because it just interacts and exchanges information with its immediate neighborhood due to the limited communication range. Consequently, it is important that decentralized monitoring provides detailed information about a user's vicinity, which influences its behavior the most. However, limiting the monitoring scope to a small part of the network constitutes an impractical design decision, because the state of the remaining network is not

taken into consideration for a possible adaptation of the communication network. In contrast, the provisioning of detailed information from the whole network leads to a dissipation of resources, including a device's energy and the limited transport capacity of the wireless communication medium. As a result decentralized monitoring must provide detailed information about a user's vicinity and deliver aggregated information about the remaining and distant regions of the network. We refer to this characteristic, which summarizes the fifth challenge, as *location-aware monitoring*.

1.3 RESEARCH GOALS

Given the identified challenges, we postulate our *main objective* as "the design of decentralized monitoring approaches in MANETs to provide accurate and location-aware monitoring information of the current network state". To accomplish this goal the main objective is divided into three major research goals.

Research Goal 1: *Survey and classification of approaches for decentralized monitoring.*
We examine concepts and existing approaches for decentralized monitoring to describe the basic architecture and categorize relevant components. Based on a selection of representative examples, we qualitatively analyze strengths and weaknesses and exploit these insights for the design of our targeted decentralized monitoring approaches.

Research Goal 2: *Design of decentralized monitoring approaches.*
Given the insights into concepts and existing solutions of decentralized monitoring, we design two fundamentally different approaches for decentralized monitoring in MANETs to fill in the gap of existing solutions and address our research challenges.

Research Goal 3: *Ascertain the correct mode of operation.*
To ascertain if the identified research challenges are met and, specifically, if *accurate* and *location-aware* monitoring information are provided we conduct a thorough and extensive evaluation.

As indicated by these goals the thesis primarily focuses on the design of decentralized monitoring approaches and how they operate in the presented application scenarios with respect to performance and cost. The influence of non-cooperative or even malicious users on the developed approaches is not considered in this thesis. For further information about research in the area of countermeasures against non-cooperative or malicious users in MANETs we refer the interested reader to [88, 44].

Dealing with the purpose of monitoring, we focus on solutions that belong to the class of approaches, which are used for performance management, as specified by the International Organization for Standardization [62]. In particular, this covers the provisioning of information of the current network state, including the users' devices, the interconnecting links between them, and end-to-end abstractions [96]. As a matter of fact this excludes, e.g., monitoring of private or personal information, of network attacks, or the creation of user histories.

1.4 THESIS ORGANIZATION

The rest of the thesis is structured as follows. In Chapter 2 an overview on the relevant technical background is given to understand this thesis and its contributions. We introduce transition-enabled communication networks, which constitute a new paradigm for communication networks, as envisaged by the Future Internet project MAKI [49]. The detailed description is essential to understand the importance of monitoring in this class of communication networks. The monitored data and gained information about the current state of the network serve as basis for potential adaptations of a transition-enabled communication network. Besides the overview on communication networks, we present the technical foundations of MANETs as the considered class of communication networks and describe the discussed application scenario. Furthermore, an introduction to decentralized monitoring is given, addressing the foundations and requirements of centralized and decentralized monitoring in communication networks. Chapter 3 deals with the examination and classification of the related work in the area of decentralized monitoring. For this survey we broaden our view of decentralized monitoring and additionally include approaches for other types of communication networks to widen the scope for the performed examination and classification.

In Chapter 4 we present our first solution for decentralized monitoring in MANETs, denoted as BlockTree.KOM. BlockTree.KOM operates on a hierarchical topology, which serves for the data exchange and is exploited to provide location-aware monitoring information. We start with a description of the concepts for the hierarchy and for the data exchange over the hierarchy. Given the concepts, we present two approaches, referred to as C-BlockTree.KOM and P-BlockTree.KOM, which rely on these concepts but implement them in different ways.

With Mobi-G.KOM we introduce our second solution for decentralized monitoring in MANETs in Chapter 5. Mobi-G.KOM operates on a flat topology and uses the robust communication pattern of gossiping to exchange information. Without an underlying hierarchy Mobi-G.KOM sacrifices the functionality of providing location-aware monitoring information. In contrast, it must not establish and maintain the underlying hierarchy.

After the presentation of our contributions, Chapter 6 deals with the evaluation of BlockTree.KOM and Mobi-G.KOM. The evaluation consists of two separate parts, comprising (i) the examination of relevant system parameters and their influence on performance and cost and (ii) a comparative evaluation of the two monitoring solutions. Finally, Chapter 7 (i) summarizes this thesis, (ii) highlights our contributions and major findings, and (iii) concludes with an overview about future work and next steps to push decentralized monitoring to the next level.

2 BACKGROUND

The main objective of this chapter is to provide an overview on the related and relevant technical background. This background eases the understanding (i) of the arising requirements on decentralized monitoring and (ii) of the design decisions for our developed solutions for decentralized monitoring. During the overview we describe relevant technical aspects and concepts, state general assumptions, and specify basic terms, which are used throughout the thesis.

Figure 1 depicts the structure of this chapter. It sketches the embedding of the communication network into the application scenario and highlights relevant layers that are discussed in the following. In Section 2.1 we start with an overview on the application scenario that encloses and influences the considered communication network. During this description we detail the characteristics of the environment as well as of location-based services, which constitute our choice as the corresponding application example. Given the description of the enclosing setup, we introduce communication networks in Section 2.2 with a focus on transition-enabled communication networks. Afterwards, Section 2.3 deals with an overview on MANETs, which represent the considered class of a communication network in this thesis. In this section we particularly focus on the network layer with the related concepts to route data from one to another or multiple nodes. To complete this chapter Section 2.4 introduces decentralized monitoring of communication networks. After a common specification of monitoring in communication networks, we describe the structure of decentralized monitoring mechanisms and identify relevant functional and, most important, non-functional requirements.

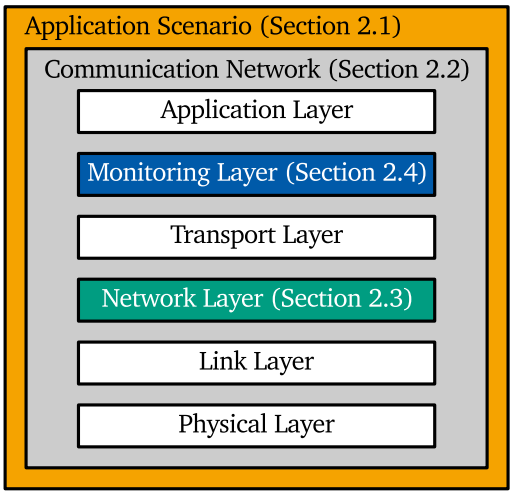


Figure 1: Overview on the structure of the chapter with the detailed layers of the communication network. The depicted architecture of the communication network represents a modified version of the reference model for computer networks from Tanenbaum and Wetherall [167].

2.1 APPLICATION SCENARIO

We start with an introduction to the application scenario, because it encloses and influences the communication network. The description details the surrounding environment and discusses suitable applications and services for the deployment in MANETs. As discussed in Chapter 1 MANETs are applicable in diverse scenarios where the direct interconnection of users may be exploited. Typically, MANETs are often considered in scenarios where an infrastructure is missing or not accessible. Examples cover first response or disaster relief scenarios [117, 44, 107] with a crashed or unavailable communication infrastructure. In addition to the deployment in these exceptional situations MANETs are applicable in ordinary situations. They may (i) offload cellular networks or (ii) serve as an alternative communication substrate for the direct interaction between nearby users. Examples, belonging to the first category, comprise data-intensive applications for real-time entertainment. They range from the distribution of content over nearby users [54, 6] to collaborative video streaming applications [80] and may exploit the local popularity of content, which is distinctive for that area and differs from the global popularity [188]. Examples of the second category, where MANETs serve as an alternative communication substrate, comprise mobile social applications. In these applications nearby users rely on the direct interconnection of their devices to exchange text messages, pictures, or small videos [97, 134, 141]. Furthermore, location-based services exploit MANETs to create, store, and replicate content that relates to nearby places and localities of a user [19, 126].

In the considered scenario of the thesis we particularly focus on location-based services [178, 153, 74, 32]. In general, they are characterized by their *locality of interaction*, since users mainly consume and produce information related to their vicinity or interact with nearby users to exchange data. In this context a MANET constitutes an useful communication substrate, because it enables the direct exchange of data without the need for a cellular network. Furthermore, location-based services do not require a MANET to route data over long distances. Instead, the communication substrate mainly provides the user with data from nearby places. We describe these characteristics in more detail in Section 2.1.2.

2.1.1 *Description of the Environment*

The description of the environment covers the envisaged location and details the mobile communication devices that are carried by the users. In our scenario we focus on *urban areas* that represent smaller or larger regions in cities. The area is populated with users, which are equipped with mobile communication devices. Users may (i) enter or leave the considered area or (ii) switch their portable devices on or off, which results in *churn*, as specified by Stutzbach and Rejaie [165]. Considering the mobility of users, we focus on *pedestrians* that walk by foot. As stated during the description of our research goals in Section 1.3 we assume a *protocol-compliant* and *cooperative user behavior*.

In terms of the considered mobile communication devices we assume that users are carrying smartphones or tablet PCs. The devices typically offer different interfaces to communicate, which range from near-field communication or Bluetooth over

WLAN to mobile broadband access. For the information exchange between the participating devices in a MANET we rely on the WLAN interface. The corresponding protocol, which implements the communication over that interface, belongs to the IEEE 802.11 protocol family [60]. It enables the direct interconnection of multiple devices using its offered ad hoc mode. For the remainder of this thesis we refer to WLAN and the corresponding protocol family as *Wireless Fidelity (Wi-Fi)* and to its offered ad hoc mode as *Wi-Fi ad hoc*, respectively. In addition to the communication interface we assume that a device relies on the *Global Positioning System (GPS)* [42] to localize its current position. Finally, the devices are *battery powered*.

2.1.2 *Description of Location-Based Services*

After the description of the environment we introduce location-based services. They represent the considered class of applications and services for the deployment in MANETs. We do not detail how related applications or services are set up and perform in a MANET, because this is beyond the scope of this thesis. Instead, we introduce location-based services and describe their characteristics to justify our choice as the corresponding application example.

2.1.2.1 *An Introduction to Location-Based Services*

In literature location-based services are defined as services, which depend on a user's current position and exploit this information to provide related information and content [178, 153, 74, 32]. They are roughly divided into *location-tracking* and *position-aware services* [74, 186]. Location-tracking services provide information about the current location of one or several entities to interested third parties, whereas position-aware services provide the user with location-dependent information [74].

For our application scenario we focus on position-aware services. Typical examples comprise but are not limited to

- the rating of nearby locations, as in Foursquare[1] or Yelp[2];
- participatory sensing applications, such as da_sense[3], where users capture information about their vicinity using the integrated sensors of their smartphones;
- crowd sourcing applications, such as AppJobber[4], where users seek for nearby mini-jobs that are rewarded by the employer on completion.

2.1.2.2 *Characteristics of Location-Based Services*

In the following we present the characteristics of location-based services and use them to justify our choice as the corresponding application example. The choice of location-based services is primarily attributed to their popularity during the last years and their applicability for the deployment in MANETs.

The popularity of location-based services is tightly coupled to the success and technological progress in the area of mobile communication and the corresponding

1 `https://de.foursquare.com` (Last accessed on July 18, 2014))
2 `http://www.yelp.de` (Last accessed on July 18, 2014)
3 `http://www.da-sense.de` (Last accessed on July 18, 2014)
4 `http://www.appjobber.de` (Last accessed on July 18, 2014)

devices [178]. It began in 1996 with the E911 mandate from the US government that required telecommunication operators to locate emergency calls in their mobile networks [153, 74, 32]. In 2005 (i) the rise of the Web 2.0 paradigm, (ii) the distribution of GPS-capable mobile devices, and (iii) the introduction of the Universal Mobile Telecommunications System (UMTS) have been responsible for the increase in popularity of location-based services, as identified by Bellavista et al. [10]. Today, as revealed by a recent study [186], 74 % of smartphone owners in the US with an age older than 18 years use location-based services.

The applicability of location-based services for the deployment in MANETs results from the locality of interaction of location-based services, as discussed in the following. Based on measurements of the location-based service Foursquare, Noulas et al. [122] show that 20 % of consecutive checkins are below a distance of one kilometer, while 60 % range between one kilometer and 10 km. Another measurement study of Foursquare outlines that 34.5 % of all users have a radius of gyration below 10 mi. (≈ 16.09 km) [22]. The radius of gyration represents the standard deviation of the distance between the locations of checkins and the average over these locations. Taking the distance between the location of users and the requested information into consideration, we analyzed data from the crowd sourcing application *AppJobber*, which was provided by the "wer denkt was GmbH". The results reveal that approximately 58.3 % of all users are interested in information that is less than one kilometer away. Only 6.4,% of all users are interested in information, which is more than 10 km away, reaching a maximum distance of 201.4 km. For further details on our data analysis, we refer the interested reader to Appendix A.1.

The provided results underpin that users are mainly interested in their vicinity and request information from nearby places, regions, or localities. Consequently, a MANET provides a viable communication substrate. The direct interconnection of devices appropriately reflects the locality of interaction of location-based services.

2.2 AN INTRODUCTION TO COMMUNICATION NETWORKS

Before we take a closer look at relevant layers of the reference model for communication networks, as depicted in Figure 1, we start with an introduction to communication networks. Afterwards, we present the notion of *transition-enabled communication networks*, as specified by the Future Internet project *MAKI* [49]. MAKI targets at the development of flexible communication networks that adapt (i) to the current conditions of the network and the environment as well as (ii) to the requirements of the application scenario.

According to the general definition from Tanenbaum and Wetherall [167], communication networks represent a collection of autonomous communication devices, which are interconnected to enable the communication between each other. To exchange information between any devices communication networks consist of multiple layers, which are vertically aligned. Through a well-defined interface, every layer offers its services to the layer above, whereas layers at the same level on different devices rely on *communication protocols* for the virtual data exchange between them. The protocols that are deployed at the different layers of the communication network form the *protocol stack*. Typical examples for protocol stacks cover the ISO OSI Reference model [31], the TCP/IP Reference Model [15], and the derived version from

Tanenbaum and Wetherall [167] that serves as reference in this thesis and is shown in Figure 1. For this thesis, the reference model has been extended with a dedicated layer for monitoring, as discussed in Section 2.4.2.

MAKI extends the presented concepts by the notion of the aforementioned transition-enabled communication networks [49]. The major idea is the development of adaptive communication networks that are able to handle the increasing dynamics. They originate from but are not limited to (i) new protocols, (ii) heterogeneous communication devices, (iii) changing conditions in the network as well as the surrounding environment, and (iv) diverse requirements of the application scenario. To react on the increasing dynamics MAKI exploits the variety and diversity of already existing communication protocols and pursues the idea of a modified communication network that eases the integration of prospective protocols. In the following we present major concepts and terms of transition-enabled communication networks.

In the context of MAKI [49] a *mechanism* serves as a representative for the existing protocols at the different layers. These mechanisms are used to build and configure a communication network. If the mechanism is decomposable into separate components, the mechanism consists in turn of multiple mechanisms. We define two or several mechanisms as *functionally equivalent*, if they provide a comparable functionality but implement the functionality in different ways. The set of functionally equivalent mechanisms is represented by a *multi-mechanism*.

The basic assumption of MAKI is that functionally equivalent mechanisms of a multi-mechanism are targeted at different scenarios with diverging conditions and requirements. Consequently, the mechanisms behave differently and exhibit differing performance and cost profiles. MAKI exploits the vast amount of these functionally equivalent mechanisms to identify the corresponding multi-mechanisms. Upon the identified multi-mechanisms new *configurations* of the considered communication network may be created that consist of different mechanisms but still provide the same functionality. Dependent (i) on the prevailing network and environmental conditions as well as (ii) on the requirements of the application scenario, an appropriate configuration with the corresponding mechanisms may be selected.

Given the basic terms, we introduce the concept of a *transition* [49]. A transition represents a seamless and automated replacement of a mechanism by another one from the same multi-mechanism. If multiple transitions are simultaneously executed, we summarize these transitions as a *multi-mechanisms adaptation*. Both a single transition as well as a multi-mechanisms adaptation represent means to adapt the communication network.

Figure 2 depicts an excerpt of a transition-enabled communication network, visualizing the introduced terms and their interdependencies. The communication network consists of three adjacent communication layers, which comprise one multi-mechanism per layer. The given multi-mechanisms contain i, j, or k mechanisms that are used to configure the communication network. A transition between two mechanisms of a multi-mechanism is highlighted by the arrow between the two mechanisms. A multi-mechanisms adaptation is represented by the dashed arrow between the depicted system configurations.

For the seamless and automated execution of transitions a transition-enabled communication network constantly performs the following steps of an adaptation loop.

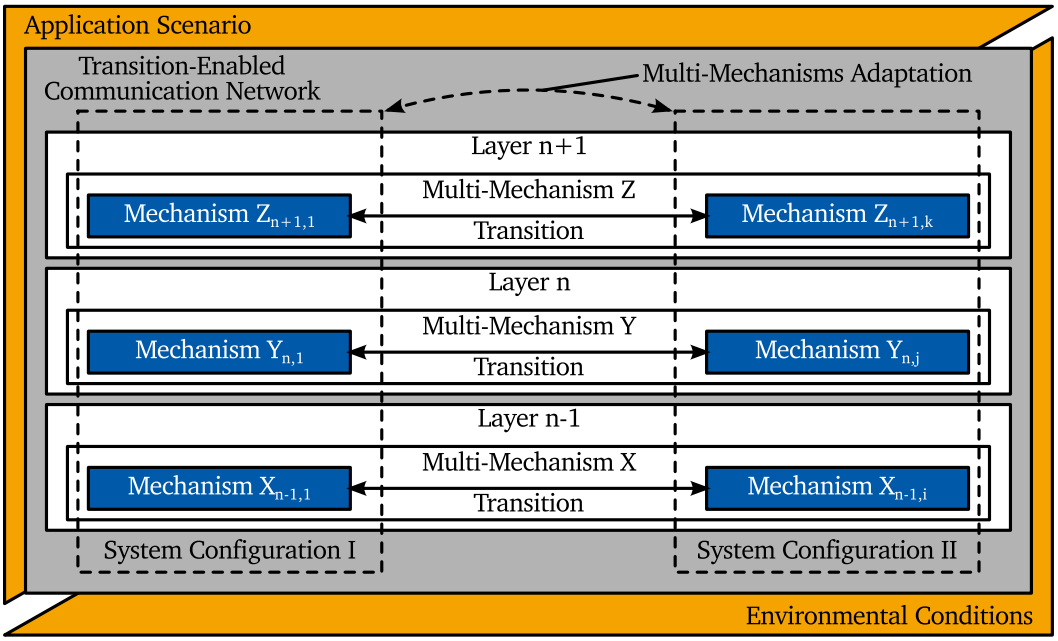


Figure 2: Excerpt of a transition-enabled communication network inspired by [49]

- The communication network is monitored to capture relevant information and to analyze the current state of the communication network and its participants, given the requirements of the application scenario and the influencing conditions of the surrounding environment.
- Based on the conducted analysis, a decision is made if transitions must be executed to adapt the communication network.
- Finally, the decision about the selected configuration is disseminated in the communication network to trigger the required transitions.

2.3 MOBILE AD HOC NETWORKS

In this section we provide an overview on MANETs and highlight common characteristics. After the introduction, we survey and classify existing routing concepts and protocols, which serve as basis and reference for the applied communication methods within our approaches for decentralized monitoring (cf. Chapter 4 and 5).

In terms of MANETs the interconnected communication devices of a communication network are mobile, meaning that they change their geographical position over time. For the remainder of the thesis we summarize the mobile communication devices as *mobile nodes* or just as *nodes* if the mobile context is obvious. In contrast to a mobile node a *fixed node* represents a node, which remains at the same geographical position over time.

MANETs are characterized as networks without an infrastructure or central point that manages the interconnection or the information exchange between nodes [144, 115, 138, 24]. Instead, the infrastructure-less network just consists of autonomous nodes, which must organize themselves to communicate with each other. The utilized communication devices are mainly battery powered, since they are mobile. As

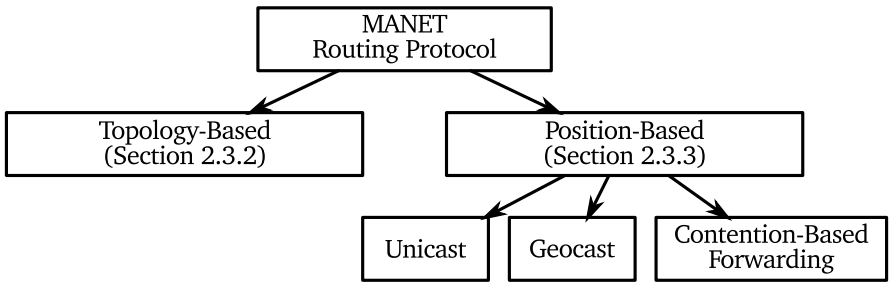


Figure 3: Overview on surveyed classes of routing protocols for MANETs

a consequence the energy of a device represents a scarce resource, which must be taken into account when designing protocols. In terms of the communication the devices communicate over a wireless and shared medium, which provides every node with a limited communication range. As the network operates on a shared medium and is neither restricted to a set of dedicated nodes nor to a certain area, every node is able to participate and to send and receive data [24, 56]. Furthermore, the wireless medium may be disrupted by environmental conditions and external signals, leading to a significantly less reliable data transmission as for fixed networks [60].

Due to the limited communication range a node just communicates with nearby nodes that currently reside inside its communication range. To address distant nodes outside of the communication range MANETs require intermediate nodes to forward the data. According to [144, 115, 56], nodes may be considered as end-systems and routers at the same time, which send and receive but also forward data. Paired with the inherent mobility and the frequently changing topology a route between a sender and a receiver only exists for a short amount of time. To enable the communication and information exchange between any two nodes dedicated routing protocols are required, which handle the mentioned characteristics, arising from the wireless communication medium and the mobility of nodes.

During the last years, a plethora of routing protocols for MANETs has been developed. We abstain from providing an exhaustive overview on existing protocols, since this is beyond the scope of this thesis. For a detailed overview on routing protocols in MANETs we refer the interested reader to [144, 162, 93, 145]. Instead, we classify existing routing protocols and give an overview about relevant and well-known approaches.

According to [115, 162, 24, 145], routing protocols for MANETs are classified as *topology-based* and *position-based* protocols, as depicted in Figure 3. In the following we detail both classes but mainly focus on position-based protocols, because they serve as basis for the communication within our approaches for decentralized monitoring.

2.3.1 *Preliminaries*

Before we detail the different classes of routing protocols and survey corresponding examples, we introduce some important terms and definitions. These terms and definitions are used throughout the thesis for a clear and consistent description of the communication between the nodes.

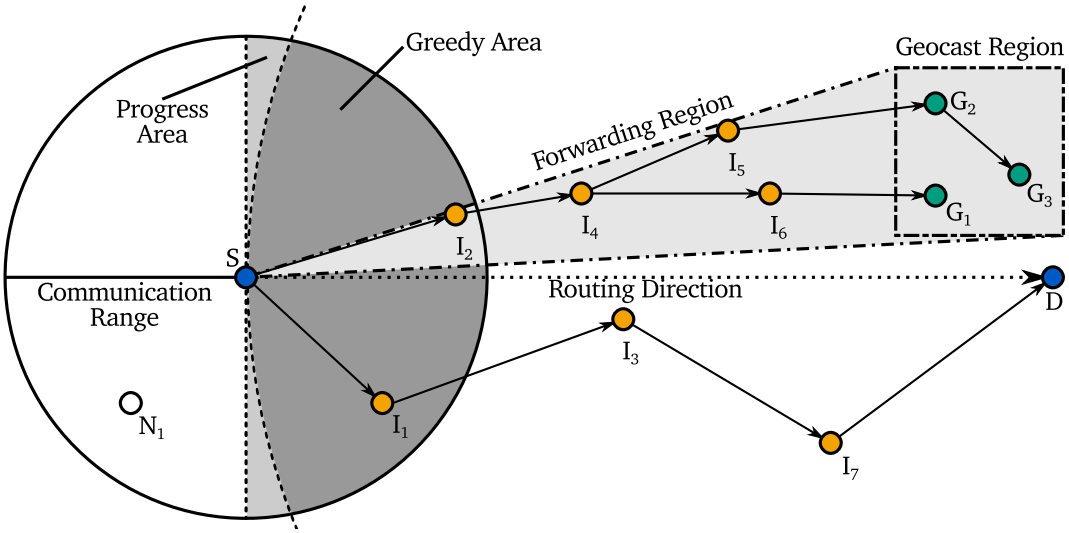


Figure 4: An example for the communication between different mobile nodes in a MANET to illustrate the different states of a node, the sets, areas, and regions

The set $\mathcal{A}$ represents all active nodes in a MANET and we refer to a single node of $\mathcal{A}$ as K. A node that intends to send a packet to another node is denoted as *source* S. The effective source of a packet as well as several potential sources are represented by the set of sources $\mathcal{S}$. The addressed receiver is called *destination* D. Since the transmission often requires several hops to reach the destination, we define a forwarding node between source and destination as *intermediate node* I. The corresponding set of intermediate nodes, which comprises all forwarding nodes between S and D is denoted as $\mathcal{I}$. A node that currently resides in the communication range of another node is denoted as a *neighbor* N and belongs to the set of neighbors $\mathcal{N}$.

Figure 4 depicts an example of a MANET with a transmission of a packet from S to D along the routing direction, which is represented by the dotted arrow. To reach the destination the packet is sent over the route $SI_1I_3I_7D$, where I_1, I_3, and I_7 represent intermediate nodes and belong to $\mathcal{I}$. Furthermore, the figure shows the communication range of S, which determines the neighbors of S, covering I_1, I_2, but also N_1.

To differentiate between multiple nodes along the route we denote the node, where a packet currently resides as current hop H^n. The exponent n represents the number of hops of a packet from S to H^n. The predecessor of the current hop, which forwarded the packet, is denoted as H^{n-1}. The successor of the current hop is labeled with H^{n+1} and belongs to the set of next hops $\mathcal{H}$. Assuming that I_4 is the current hop, as depicted in Figure 4, I_2 represents the predecessor H^{n-1}, whereas I_5 and I_6 belong to the set of next hops $\mathcal{H}$ of I_4.

In addition to the transmission of a packet to a single node a source may also transmit a packet to multiple nodes. The set of receivers may be determined by a provided list of addresses, a dedicated address, or dependent on their current geographical position. In the latter case the packet is targeted to a geographical region, which is denoted as *geocast region*. In a geocast region, the packet must be delivered to every node, which currently resides in that region. A receiving node in that region is represented by G and belongs to the set $\mathcal{D}$. To reach the targeted geocast region a packet is sent through a so-called *forwarding region*, where every

Name	Explanation
Sets	
$\mathcal{A}$	Set of all nodes K that are currently active in the MANET
$\mathcal{S}$	Set of potential and effective sources S for the transmission of the same packet
$\mathcal{D}$	Set of all receivers G of a packet, which currently reside in the geocast region. If the packet is intended for a single node, the receiver is denoted as D.
$\mathcal{I}$	Set of nodes I, which forwarded the packet from S to D or G
$\mathcal{H}$	Set of next hops H^{n+1} from the currently transmitting hop H^n, which is n hops away from the source
$\mathcal{N}$	Set of all neighboring nodes N, which currently reside in the communication range of a node
Areas and regions	
Geocast region	Defines the area, where the packet must be delivered to
Forwarding region	Represents the area, where receiving nodes become intermediate nodes and must forward the packet
Progress area	Denotes the semicircle around a sender that faces the destination or geocast region
Greedy area	Specifies the overlap between the communication range of a node K and the circle around the destination D. The radius of the circle around D is given by $\overrightarrow{DK}$.

Table 1: Summary of the nomenclature and definitions of the required sets, areas, and regions for routing in MANETs

intermediate node I must forward the message. Figure 4 depicts an example of a transmission of a packet from S through the forwarding region to the geocast region with the subsequent dissemination of the packet to G_1, G_2, and G_3.

In addition to the definition of the two regions the *progress area* specifies the semicircle around a sender towards the destination or geocast region. It is defined by the intersection of the communication range with the dashed line that is perpendicular to the routing direction $\overrightarrow{SD}$. The *greedy area* consists of a fraction of the progress area. As depicted in Figure 4 it is given by the overlap of the communication range with the dashed circle around D. The radius of the dashed circle is specified by $\overrightarrow{DS}$. Table 1 summarizes the introduced nomenclature and definitions of the different sets, areas, and regions.

2.3.2 *Topology-Based Routing Protocols*

Topology-based routing protocols for MANETs rely on the links between the nodes. Upon these links, the nodes continuously maintain routes to other nodes or establish and maintain the routes, if required. Based on these differences, topology-based routing protocols are subdivided into *proactive*, *reactive*, and *hybrid* protocols [115, 162].

In proactive routing protocols each node must maintain a routing table, which contains an entry with a distance to every other node in the network. The routes of each routing table are managed and kept up-to-date through periodical [23] and/or event-driven updates [132, 118]. Link-state-based approaches [131, 64] require nodes to disseminate their link-state information so that other nodes are able to create and maintain their routes to every other node.

In contrast, reactive protocols avoid the maintenance and management of routes to every other node in the network. Instead, a route is only created and managed if required. Therefore, the source of a packet discovers the route to the destination. To maintain the identified route intermediate nodes (i) store their successors along the route [133, 171], (ii) store every adjacent node of a resulting directed acyclic graph, which is routed at the destination [129], or (iii) the source caches the complete route to every destination [71].

Hybrid routing protocols combine aspects from proactive and reactive routing protocols. For instance, a proactive protocol may be used between nearby nodes, whereas distant nodes communicate over a reactive protocol [52].

2.3.3 *Position-Based Routing Protocols*

Position-based routing protocols do not operate on existing links between the nodes but incorporate the information about a node's geographical position to determine the route from a source to a destination. The protocols require the position of the destination as well as the positions of the nodes along the routing direction to determine the effective next hop. Exploiting the information about geographical positions, nodes neither initiate network-wide searches for a receiver nor maintain routes to potential or all destinations in the network. Instead, a node locally decides on the next hop without maintaining a cache for routes or nodes, as detailed below. Consequently, position-based routing protocols scale with the number of nodes in the network and outperform topology-based protocols in this regard, as experimentally confirmed in [101, 66].

In general, position-based routing protocols either route messages to single nodes or to multiple nodes, which currently reside in a specified region. This region corresponds to the aforementioned geocast region and the corresponding protocols are often referred to as *geocast protocols*. In the following we review position-based unicast and geocast protocols, because the underlying addressing and routing concepts play an important role for the communication within our approaches for decentralized monitoring.

2.3.3.1 *Unicasting*

For position-based unicast protocols a source requires the information about a destination's current geographical position. Furthermore, every intermediate node must know where its neighbors currently reside in order to decide on the next hop and to forward the message to the destination. Therefore, position-based unicast protocols require a *location service* and a *forwarding strategy*. The location service is used to obtain the current position of the addressed node. Due to the fact that our approaches for decentralized monitoring do not require a node's geographical position to iden-

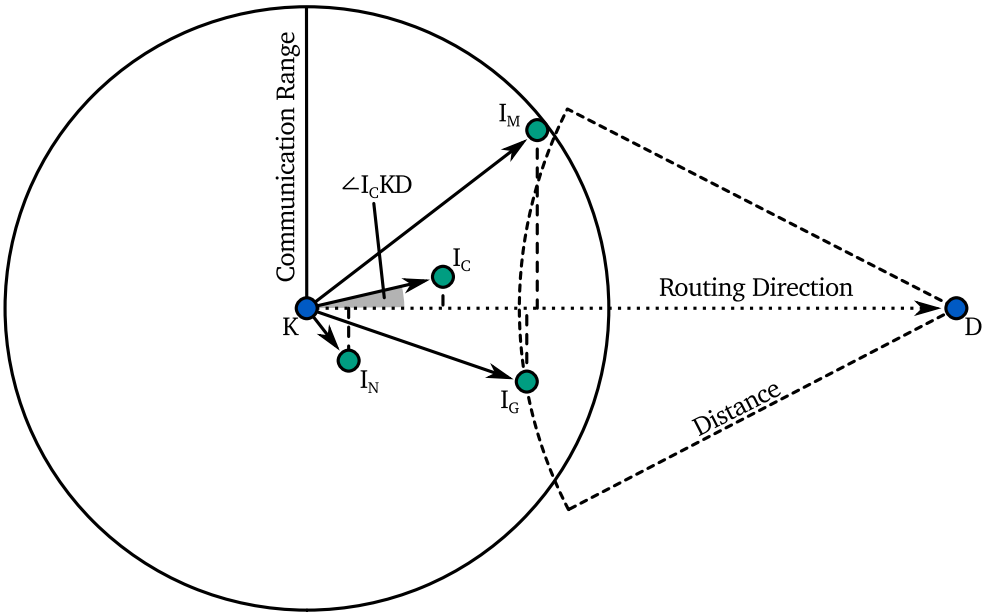


Figure 5: Next hop selection strategies for greedy routing in MANETs inspired by [162]

tify the destination of a packet we omit a survey on existing solutions and refer the interested reader to [7, 101, 115, 83, 106].

Based on the obtained information about the current geographical position of the destination, the forwarding strategy determines the corresponding route. Therefore, every node periodically broadcasts its current position [115, 162], which is used to select the next hop according to the applied *next hop selection strategy*. During each hop, the strategy tries to select a neighbor that minimizes the distance to the destination. The resulting greedy strategy fails if a so-called *concave node* [162] is reached. A concave node has no neighbors that reduce the distance to the destination. For this reason, a *recovery strategy* is applied to leave the concave node. In the following we present an extract of prominent and relevant solutions. For further details and approaches we refer the interested reader to [43].

NEXT HOP SELECTION STRATEGY

We rely on Figure 5 to support the explanation of prominent and relevant next hop selection strategies. The figure shows a sending node K, which transmits the packet to D along the routing direction. The green nodes in the communication range of K constitute potential intermediate nodes along the routing direction. Dependent on the selected next hop selection strategy, one node becomes the intermediate node I.

Several solutions rely on the progress as selection criterion for the next hop. The progress for the next hop is determined by its projection onto the routing direction (cf. Figure 5). The *Most Forward Within Radius* scheme [166] selects I_M as next hop, since it maximizes the progress. Contrary to this, the *Nearest Forward Progress* [57] chooses the nearest neighbor I_N of K to reduce the probability of collisions as well as to minimize the energy consumption for a transmission. In addition to the progress criterion the approach from Finn [39] selects node I_G, which has the smallest distance to D. Compass routing [92] selects node I_C, because it reduces the angle $\angle I_C KD$, as depicted in Figure 5.

RECOVERY STRATEGY

Multiple approaches exist to recover from a concave node. In [163] concave nodes flood their neighbors, which, in turn, proceed with the greedy forwarding strategy as long as they are no concave nodes themselves. In [166] it is proposed to return the packet backwards to the node with the least negative progress. Another class of approaches [13, 77] applies face routing to leave concave nodes and to bypass empty regions. If an intermediate node is reached that is closer to the destination than the concave node, the node proceeds with the next hop selection strategy.

2.3.3.2 *Geocasting*

Geocast protocols are classified into *data-transmission oriented* and *routing-creation oriented* protocols [70, 110]. The first class of protocols transmits the packets from the source to the geocast region by determining the forwarding region between the source and the geocast region. Nodes inside the forwarding region become intermediate nodes and forward the packets to the geocast region. Different approaches exist to determine the forwarding region and to reduce the number of forwarding nodes to a minimum. Examples cover the logical partitioning of the area between the source and geocast region into squares [85, 103] or Voronoi regions [164].

Contrary to the first class of geocast protocols, the routing-creation oriented protocols [87, 12] create routes from the source to the specified region to avoid the well-known problem of so-called broadcast storms [172]. In a broadcast storm every receiver of a broadcast starts a broadcast on its own, which leads to a congestion of the wireless communication medium. Once the routes are established, the packets are sent to the geocast region.

2.3.3.3 *Contention-Based Forwarding*

So far, the presented position-based routing protocols require that a node is aware of the geographical positions of its neighbors to forward a packet. As outlined in Section 2.3.3.1 nodes periodically broadcast their current position so that a node is able to select its appropriate next hop. These position updates may lead to the already mentioned broadcast storm problem [172], which congests the wireless communication medium. Moreover, if no or only a few packets are routed between sources and destinations, periodic updates are unnecessary, because recent position updates are not required.

Contention-based forwarding schemes [40, 189], also known as *beacon-less routing* [53], present an efficient extension of position-based routing protocols, since periodic updates are completely avoided. Instead, a forwarding node broadcasts a packet and each receiving node, which resides in a specified area, becomes a candidate for the next hop. Shape and size of the area depend on the corresponding contention-based forwarding scheme [40, 189, 53]. For example, the area may consist of the greedy area (cf. Figure 4). To determine the effective next hop every receiving node starts a timer. The length of the timer depends on (i) the position of the next hop, (ii) the next hop selection strategy, and (iii) the position of the previous hop or the destination. Assuming that the Most Forward Within Radius scheme [166] is used, nodes inside the greedy area with a higher progress obtain a smaller timer than nodes with a lower progress. If a timer expires, the corresponding node broadcasts the packet. The remaining waiting nodes receive this packet and cancel their delayed transmission of

the packet. For contention-based forwarding schemes it is essential that nodes are able to overhear transmitted packets from other nodes.

2.4 DECENTRALIZED MONITORING IN COMMUNICATION NETWORKS

Subsequent to the introduction of MANETs we provide the background on decentralized monitoring. At the beginning, we start with the foundations of monitoring in communication networks and survey different classes of network monitoring. After bridging the gap to decentralized monitoring, we detail general principles of decentralized monitoring, covering the structure of a corresponding mechanism and a detailed description of its relevant components. Finally, we present the non-functional requirements of a decentralized monitoring mechanism.

2.4.1 *Foundations of Monitoring*

The Oxford dictionary defines the verb *to monitor*, as "observe and check the progress or quality of (something) over a period of time" [33]. In terms of communication networks monitoring is described as continuous measuring, collecting, interpreting, and displaying of information of objects [72, 113, 34]. These objects may represent concurrent processes on a single node [72] or processes on different nodes in communication networks [34]. For the remainder of the thesis and to comply with the concepts of mechanisms, as introduced by MAKI [49], we refer to the service that monitors a communication network as *monitoring mechanism*. The monitored state of the network and its participating nodes is denoted as *monitoring information* and represented by a set of *attributes* [113].

In a communication network monitoring information provides the data basis to manage the system and to react on changes. These changes may arise from the network and its participating nodes [113, 96] as well as from the changing conditions of the surrounding environment and application requirements, as specified by MAKI [49]. The class of monitoring with the related monitored attributes depends on the aspects of a communication network, which should be managed. The International Organization for Standardization has identified five main classes of network management [62], also known by the abbreviation *FCAPS*, that specify which aspects must be monitored and managed.

FAULT MANAGEMENT comprises the detection of faults, the identification of appropriate countermeasures, and the execution of the selected actions. Monitoring provides the data basis to detect these faults and to determine the corresponding severity.

CONFIGURATION MANAGEMENT describes the task to maintain the network by controlling the configuration of every participating node. For this purpose monitoring is used to capture changing configurations. The obtained information may serve for the identification of potential networking issues, which are attributed to incorrectly configured nodes.

ACCOUNTING MANAGEMENT deals with the compliance of access rights and of resource utilization in a communication network. In this context monitoring captures, which resource has been accessed to which extent. The collected data

may be used to verify the correctness of access rights over time and/or for billing dependent on the resource utilization.

PERFORMANCE MANAGEMENT addresses the performance of communication networks and examines if the specified requirements are met. In this regard monitoring is used to measure the current state of the network, comprising the participating nodes, the interconnecting links between neighbors, and end-to-end abstractions between distant nodes [96]. The provided data serve as basis to identify (i) potential bottlenecks, (ii) the availability of sufficient resources, or (iii) the dissipation of unused resources.

SECURITY MANAGEMENT deals with the detection of attacks and identification of infected or malicious nodes in the communication network. In this context a monitoring mechanism may be used to monitor the traffic, create traffic profiles, and send alerts if unusual or suspicious traffic is detected.

Among the different management and monitoring classes, we primarily focus on monitoring mechanisms that belong to the class of *performance management*. The considered mechanisms monitor a predefined set of attributes to represent the state of the network and its participating nodes. As envisioned by MAKI [49] and explained in Section 2.2 the monitoring information is used to analyze the communication network and to decide on potential transitions of the communication network.

Given the foundations of monitoring, we derive the functional requirements and specify that a monitoring mechanism must

- observe the quality and performance of the communication network, by
- continuously measuring, collecting, and interpreting monitoring information, which are represented by a set of specified attributes that
- reproduce the state of the communication network and its participating nodes.

So far, the description about monitoring and management implicitly assumes the presence of a central monitoring entity. Figure 6 sketches an example of a monitoring mechanism with a central monitoring entity that monitors the communication network. On top of the lower four layers of the communication network, which are summarized as *underlay*, the central monitoring entity is responsible to monitor the network by collecting and processing the data from the nodes. In this regard *CoMon* [128] represents a prominent example for a monitoring mechanism with a central monitoring entity. CoMon monitors the testbed environment PlanetLab and consists of a central node that periodically requests a set of attributes from the remaining nodes in the network. Subsequently, the central node processes the data and offers insights into the current network state.

Depending on the number and distribution of nodes in the network, the central monitoring entity may also consist of multiple monitoring nodes, which are transparent to the remaining nodes in the network. Thus, the depicted central monitoring entity in Figure 6 may be considered as substitute for multiple monitoring nodes. The *Query* infrastructure from Akamai represents an example that belongs to this category, because it consists of more than 60,000 nodes, which are distributed over the whole world [140]. Query operates on a hierarchical architecture, which facilitates the collection of data at single and multiple locations, thus, exhibiting aspects

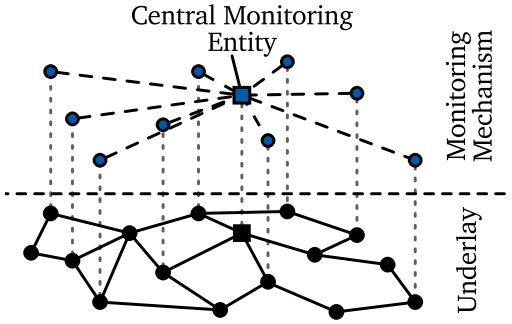


Figure 6: Example of a monitoring mechanism with a central monitoring entity that monitors the communication network

of a centralized but also decentralized design. It monitors the Akamai network for different types of large-scale applications and collects data from the own servers, the network, users, and agents[5] [123].

Figure 6 and the examples of CoMon and Query illustrate and emphasize the necessity of a central monitoring entity. However, the considered communication networks of this thesis are exclusively established by the nodes and are not equipped with a central entity to monitor the communication network. As mentioned beforehand related examples of these communication networks comprise but are not limited to MANETs and P2P networks. Similar to MANETs (cf. Section 2.3) nodes of a P2P network organize themselves to share and provide their resources for the operation of the communication network [3, 154]. Corresponding examples for P2P networks range from (i) mobile P2P networks in MANETs [148, 184, 45], over (ii) hybrid networks [58, 108], interconnecting mobile and fixed nodes, to (iii) solutions, which are tailored to fixed nodes [161, 143, 90]. Due to the distributed nature of the presented communication networks without a central entity the nodes cannot rely on a central monitoring entity. Instead, they must operate the deployed monitoring mechanism by themselves to monitor the communication network.

2.4.2 *Principles of Decentralized Monitoring*

After introducing the foundations of monitoring in communication networks we present the principles of decentralized monitoring. These principles serve for the design of appropriate decentralized monitoring mechanisms that comply with the requirements, which have been derived and presented during the previous section. To be more precise we answer the following questions about decentralized monitoring mechanisms:

1. What is the basic design of a decentralized monitoring mechanism?
2. How is a decentralized monitoring mechanism integrated into a communication network?
3. Which tasks have to be performed by the nodes so that data are collected and that the generated results are subsequently disseminated in the network?

5 An agent simulates a requesting user to assess the performance of the request and the quality of the requested content.

To answer these questions we present the overall structure of a mechanism for decentralized monitoring in Section 2.4.2.1. Afterwards, we go into detail and describe relevant components of the mechanism in Section 2.4.2.2 and 2.4.2.3. Finally, Section 2.4.2.4 provides a summary on the principles of decentralized monitoring, which are not limited to decentralized monitoring mechanisms in MANETs but address communication networks in general. We point out, if some of the statements are only applicable to a certain class of communication networks.

2.4.2.1 *The Structure of Decentralized Monitoring Mechanisms*

A major task of monitoring is the measurement and collection of monitoring information, which consists of a set of attributes to reflect the current state of the communication network and its participating nodes (cf. Section 2.4.1). Correspondingly, decentralized monitoring mechanisms must execute this task as well. Due to the fact that decentralized monitoring mechanisms are deployed in a communication network without a dedicated monitoring entity the participating nodes are responsible to measure and collect the data. For the remainder of this thesis we refer to this task as *data collection*, representing the first basic component of a decentralized monitoring mechanism. In the context of data collection we denote the collected data as *monitoring data*.

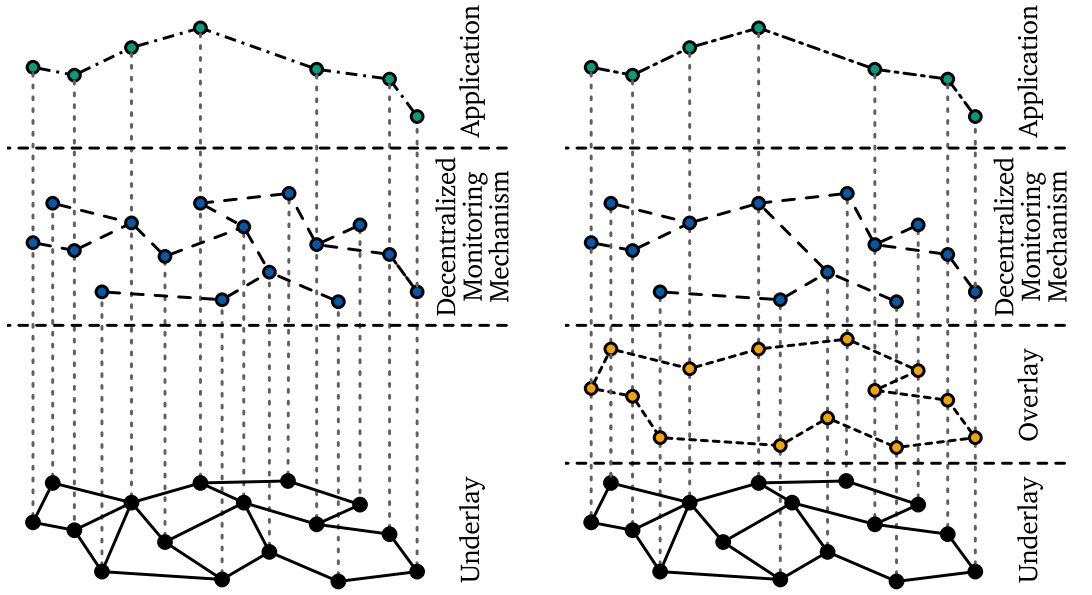
Subsequent to the collection, the information about the current network state must be disseminated among the nodes. There is no central entity that exploits and utilizes monitoring information to manage the communication network. Consequently and in addition to the collection of monitoring data, the nodes must be provided with the monitoring information to manage communication networks. We denote the subsequent dissemination of monitoring information as *result dissemination* and refer to the disseminated information as *monitoring results*. The result dissemination represents the second basic component of a decentralized monitoring mechanism.

For the collection and dissemination a dedicated topology must be established and maintained. The topology specifies on which nodes the data are collected and how a single or multiple collecting nodes are reached. With respect to the dissemination of monitoring results either a separate or the same topology may be used to specify how monitoring results are disseminated. We denote the underlying topology for the collection and result dissemination as *monitoring topology*. The monitoring topology incorporates the tasks to establish and maintain the topology and constitutes the last basic component of a decentralized monitoring mechanism.

Based on the discussion, we are able to answer the first of our three questions that address decentralized monitoring mechanisms. As presented in our previous work [156, 158] the structure of a decentralized monitoring mechanism consists of three basic components, covering (i) the monitoring topology, (ii) data collection, and (iii) result dissemination. In the following we detail the concepts of each component, while existing solutions for a decentralized monitoring mechanism with its basic components are surveyed in Chapter 3.

2.4.2.2 *Monitoring Topology*

In order to answer the second from our three questions we elaborate on the integration of a decentralized monitoring mechanism into the communication network.



(a) Direct integration of a decentralized monitoring mechanism on top of the underlay

(b) Integration of a decentralized monitoring mechanism on top of an overlay

Figure 7: Integration of a decentralized monitoring mechanism into different communication networks

Afterwards, we focus on the monitoring topology and the related tasks and arising requirements. We integrate a decentralized monitoring mechanism on a separate layer into the considered communication network above the transport layer. At this layer, the decentralized monitoring mechanism is able to access and use the routing functionality of the underlying layers to route data through the network to a given destination. Depending on the considered communication network, a decentralized monitoring mechanism may directly use the routing functionality from the transport layer or from an additional layer denoted as *overlay layer*, which is in between the transport and monitoring layer. Both alternatives are depicted in Figure 7.

Overlays are used for the logical interconnection of nodes. They provide unified methods to communicate and abstract from the complexities of the network [30], comprising different access techniques [20] or addressing schemes [58, 59]. Especially in P2P systems, overlays constitute a well-known concept. Besides the provisioning of means to communicate, they are used to lookup information [161, 90], distribute content [26, 78, 136], stream videos [187, 179, 1], or implement even decentralized accounting schemes [104]. Figure 7a depicts an example, where the decentralized monitoring mechanism is directly integrated on top of the underlay. The provided addressing and communication schemes from the underlay serve for the establishment of the monitoring topology. Figure 7b shows an example of a decentralized monitoring mechanism on top of an overlay using the functionality from the overlay to establish and maintain the topology.

As mentioned above the monitoring topology is used to collect data, which are distributed over the whole network and are available at the participating nodes. The topology identifies one or multiple nodes to store the collected data and specifies how the data must be forwarded to reach one or several of these particular nodes. For

the dissemination of monitoring information the same or an additional monitoring topology may be used.

Once established, the maintenance of the topology is of major importance, because it provides the basis for the exchange of monitoring information. Consequently, the monitoring topology must deal with the characteristics and dynamics that arise from the MANET and the surrounding environment. As specified by the second and third research challenges and exemplified in the description of the environment (cf. Section 2.1.1) the monitoring topology must handle (i) the constant arrival and departure of nodes, also referred to as churn, (ii) the mobility of nodes in MANETs, and (iii) the limited communication range of the devices in those networks, which leads to a constantly changing network topology in combination with node mobility.

2.4.2.3 *Data Collection and Result Dissemination*

As introduced in Section 2.4.1 the monitoring information about the network and its participating nodes is represented by a set of attributes. Existing approaches enable the adding or removal of attributes at runtime [176]. In contrast to that we define the set of attributes as a fixed set, which cannot be changed at runtime. The set may consist of attributes from different communication layers but also take the device itself into consideration, as shown by the following examples.

UNDERLAY. Examples of attributes comprise the amount of sent and received data of a node [114], the packet loss rate [130], or the expected transmission time [35] to characterize the state of the routing protocol.

OVERLAY. The number of participating nodes constitutes a frequently selected attribute for the overlay, which is monitored to adapt the structure of the overlay [112, 169] or to adjust the join and leave procedures of the overlay [2].

DECENTRALIZED MONITORING MECHANISM. As proposed in [102] a decentralized monitoring mechanism captures the request rate of monitored data. Based on the obtained insights, the process of data retrieval may be adapted in order to reduce latency and cost.

APPLICATION. Examples for monitored attributes at the application layer are highly diverse. For instance, (i) the access frequency of content is monitored to adjust content replication [27, 99] or (ii) social relations between nodes are captured to place content on nodes with strong social relations [127].

COMMUNICATION DEVICE. In addition to layer-specific attributes resources and characteristics of the communication devices may also be monitored [46, 97]. Examples cover the available storage, the energy state, or existing CPUs.

After the overview on monitored attributes of a decentralized monitoring mechanism, we answer the last of our three questions. We focus on data collection and result dissemination discussing related tasks and arising requirements.

DATA COLLECTION

Every participating node of a decentralized monitoring mechanism must locally measure the set of specified attributes. Afterwards, the attributes are forwarded over the

topology and collected at a single or multiple nodes dependent on the topology. During the data collection phase, the measured attributes are processed primarily to reduce the size of the collected data. The reduction is necessary to prevent that the resulting traffic becomes the predominant traffic factor and exceeds the traffic of an application or service. In this context aggregation represents a frequently used procedure to compress the size of data. During the collection phase, the measured attributes and/or partial results, which represent already combined attributes from two or more nodes, are aggregated. Examples of common aggregation functions for the combination of monitored attributes comprise minimum, maximum, sum, count, or average [109] but also central moments (e.g. variance [69]) or histograms with accuracy objectives [75]. The presented process of data aggregation during the transmission is often referred to as *in-network aggregation* [135].

Independent of the processing technique, the final result of an attribute contains the input from all participating nodes that are covered by the decentralized monitoring mechanism. According to the nomenclature from our previous work [158], we denote the final result of an attribute, comprising the input from all nodes, as *global view* of an attribute. The monitoring results contain the global views of all monitored attributes representing the *global state* of the communication network.

In contrast to decentralized monitoring mechanisms for fixed communication networks approaches for MANETs must not necessarily integrate the measured data from all nodes that reside in the network. The limited communication range paired with the distribution of nodes over a large geographical area leads to a high network diameter of the MANET. Given this situation, the integration of data from distant nodes leads to an unnecessary dissipation of a node's resources. Consequently, decentralized monitoring mechanisms for MANETs limit their scope to a predefined area. On the one hand, this area exceeds the local view of a single node to encompass more nodes and provide substantial information about the communication network, as demanded by our research challenges (cf. Section 1.2). On the other hand, the size of this area is limited to prevent the dissipation of resources. Dependent on the scope of the decentralized monitoring mechanism and the spatial network size, only a fraction of the whole network is covered. To monitor the whole MANET multiple instances of a decentralized monitoring mechanism may be distributed over the MANET. They operate in parallel without exchanging information between them.

As demanded by our fifth research challenge (cf. Section 1.2) a decentralized monitoring mechanism in MANETs must provide location-aware monitoring information. This comprises detailed information about a node's vicinity and summarized information from distant places. The requirement of location-aware monitoring reflects a typical and inherent interaction pattern in MANETs, because a node must always interact with its immediate neighbors even when interacting with distant nodes. Examples comprise (i) location services for position-based routing protocols [7, 101], where position information about nearby nodes is more accurate than about distant nodes or (ii) proactive topology-based routing protocols [131], where nearby nodes frequently update their link state information as opposed to distant nodes. Furthermore, the locality of interaction represents the typical interaction pattern of location-based services, as outlined during the description of the considered application scenario (cf. Section 2.1).

RESULT DISSEMINATION

Subsequent to the collection and processing of monitoring data, a decentralized monitoring mechanism must provide the monitoring results. Based on this dissemination, every node in the communication network is able to access and utilize the information. The information may be disseminated over the same topology, which is used for data collection. Otherwise, a separate topology serves for the dissemination of monitoring results. Furthermore, interested nodes may actively request the monitoring results from one or multiple nodes, which store the collected data.

2.4.2.4 *Summary on Decentralized Monitoring Mechanisms*

After the discussion about principles of decentralized monitoring, we summarize our findings for a corresponding decentralized monitoring mechanism that observes the quality and performance of the network to reproduce the state of the network and its participating nodes. Based on our specification of a monitoring mechanism (cf. Section 2.4.1), we postulate that a decentralized monitoring mechanism

- operates on top of the transport layer or overlay layer and exploits the functionality of the underlying layers to exchange data;
- consists of three components for (i) the monitoring topology, (ii) the data collection, and (iii) the result dissemination;
- must not necessarily cover the whole communication network (in terms of MANETs), but requires that each covered node participates in monitoring.

Consequently, every participating node (i) helps to establish and maintain the monitoring topology, (ii) locally measures and stores monitoring data, (iii) forwards or collects the monitoring data dependent on the topology, (iv) processes the monitoring data during the collection phase, and (v) disseminates the monitoring results in the network, so that every participating node is aware of the current network state.

2.4.3 *Non-Functional Requirements of Decentralized Monitoring Mechanisms*

Based on our specifications of a decentralized monitoring mechanism, we formulate the corresponding non-functional requirements to complete the chapter. We adopt the nomenclature from Saller et al. [146], who surveyed relevant non-functional requirements for fixed communication networks and divide these requirements into *workload-independent* and *workload-dependent non-functional requirements*. The class of workload-independent non-functional requirements directly characterizes a decentralized monitoring mechanism. In contrast, the workload-dependent non-functional requirements describe the influences, a decentralized monitoring mechanism must handle. The resulting identification and classification of non-functional requirements is based on our previous work for decentralized monitoring mechanisms in fixed communication networks [156, 158] and MANETs [157].

2.4.3.1 *Workload-Independent Non-Functional Requirements*

The workload-independent non-functional requirements are used to characterize a decentralized monitoring mechanism in terms of *performance* and *cost*. As detailed

below the performance is further divided into *accuracy* and *timeliness*. Based on the identified set of non-functional requirements corresponding metrics may be derived to quantify to which extent the non-functional requirements are met. Table 2 provides an overview on relevant metrics for the respective non-functional requirements.

ACCURACY

A decentralized monitoring mechanism tries to reproduce the effective network state as precise as possible. This includes the integration of information from all participating nodes that are covered by the decentralized monitoring mechanism. Thus, the accuracy of a decentralized monitoring mechanism is divided into the error [9, 89, 47, 158] and completeness [51, 18]. The error characterizes the deviation of the monitored global view from the effective global view of an attribute, whereas completeness assesses how many nodes contributed with their local values. Furthermore, the accuracy serves to validate if the functional requirement of location-aware monitoring is met.

TIMELINESS

In addition to the provisioning of an accurate global network state a decentralized monitoring mechanism must keep the time as minimal as possible to provide each node with the monitoring results. Thus, a decentralized monitoring mechanism must take care of a fast collection and dissemination. Timeliness characterizes how long it takes a mechanism to (i) collect the data, (ii) calculate the global view, and (iii) distribute this view among the nodes [9, 47, 157]. Consequently, the non-functional requirement indicates if the monitoring mechanism provides a recent or stale view of the network state.

COST

According to Keshav [82] and Bawa et al. [9], the arising cost of a decentralized monitoring mechanism are dividable into three categories covering (i) communication cost, (ii) computation cost, and (iii) memory consumption (Bawa et al. only consider the first two types of cost). The cost specifies how many resources must be spent to monitor the communication network and to provide the global view of the communication network. Furthermore, as emphasized by the identified research challenges (cf. Section 1.2), energy consumption constitutes an additional cost factor, which must be taken into consideration when characterizing a decentralized monitoring mechanism

Non-functional requirement	**Relevant metrics**
Accuracy	Total error [89], standard deviation of the error [89], relative error [28], completeness [51, 18]
Timeliness	Maximum age [9], average staleness [157]
Cost	Communication cost [82, 9], computation cost [82, 9], memory consumption [82], power consumption [105, 157]

Table 2: Overview on relevant metrics for the quantification of a non-functional requirement

in MANETs. The devices are battery powered so that energy becomes an important resource that may not be dissipated.

2.4.3.2 *Workload-Dependent Non-Functional Requirements*

Contrary to the previous class, workload-dependent non-functional requirements do not directly characterize a decentralized monitoring mechanism. In fact, they specify the environment, in which a decentralized monitoring mechanism must remain operating. They serve for the design of workloads to indicate how performance and cost are influenced. Table 3 provides an overview on relevant workloads for the particular non-functional requirements. The metrics of the workload-independent non-functional requirements are used to quantify the influence of an applied workload on the decentralized monitoring mechanism [146].

SCALABILITY

In terms of MANETs, the corresponding workloads of this category are used to vary the spatial size and node density of a communication network, as emphasized in Section 1.2. Given a constant density of nodes in the network, a decentralized monitoring mechanism must operate if the spatial network size decreases or increases. Given a constant spatial network size, the approach must work in the presence of a decreasing or increasing node density. The number of monitored attributes represents another aspect to characterize the scalability of a decentralized monitoring mechanism [182, 158]. It represents a workload parameter that belongs to a decentralized monitoring mechanism, whereas the two previously presented workload parameters vary aspects of the surrounding environment.

ROBUSTNESS

According to Riggio et al. [142], a decentralized monitoring mechanism in MANETs is considered as robust, if it withstands or recovers from failures and continues to provide its monitoring results. Failures of a decentralized monitoring mechanism stem from the dynamic nature of MANETs as well as from the used communication medium, covering the following three workload parameters:

- According to [114, 82, 142], churn and message loss constitute two inherent workload parameters, which address robustness and must be handled by a decentralized monitoring mechanism.
- Mobility represents another workload parameter, which leads to a constantly changing network topology with varying neighbors. It is of particular relevance for decentralized monitoring in MANETs.

Non-functional requirement	**Relevant workloads**
Scalability	Spatial network size [50], node density [61], number of monitored attributes [182, 158]
Robustness	Churn [82, 142], message loss [82, 142], mobility [180, 157]

Table 3: Overview on relevant workloads to examine a non-functional requirement

3

RELATED WORK ON DECENTRALIZED MONITORING MECHANISMS

Decentralized monitoring mechanisms rely on three basic components to monitor the system state and distribute the resulting monitoring information. As outlined in Section 2.4.2 and depicted in Figure 8, a monitoring mechanism consists of the three basic components for (i) the monitoring topology, (ii) the data collection, and (iii) the result dissemination. Following the principles for decentralized monitoring (cf. Section 2.4.2), we survey approaches for fixed communication networks in addition to existing solutions for MANETs. Based on the broader overview of the surveyed approaches, (i) we investigate if the basic components are further dividable into distinct categories and (ii) classify the surveyed approaches according to the three basic components with their identified categories. The examination of the basic components with the related classification is divided into three sections, as depicted in Figure 8. Furthermore, a fourth section additionally details commonly utilized aggregation techniques of decentralized monitoring mechanisms. At the end of this chapter, we summarize the surveyed approaches and draw our conclusions, setting the stage for the description of our developed monitoring mechanisms.

3.1 MONITORING TOPOLOGY

As outlined in Section 2.4.2 a decentralized monitoring mechanism requires its own underlying topology to collect monitoring data and disseminate monitoring results. Consequently, the selection of the underlying topology constitutes an important design criterion that influences data collection and result dissemination [111, 158]. In general, existing decentralized monitoring mechanisms are divided into *hierarchical* and *flat* approaches. Hierarchical approaches operate on top of an ordered topology,

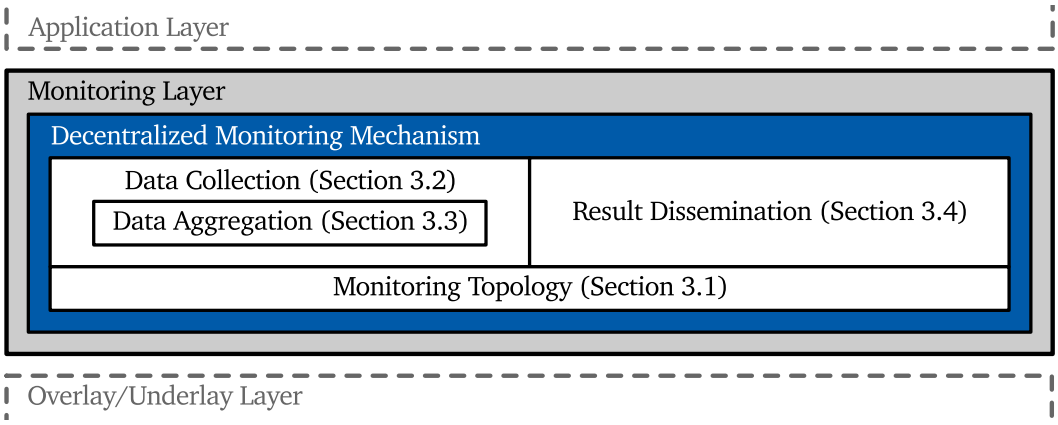


Figure 8: Overview on the three basic components of a decentralized monitoring mechanism and the resulting structure of this chapter, which contains an additional section about commonly applied data aggregation techniques

such as a tree. The monitored data is forwarded from lower to higher levels in the hierarchical topology and a single or multiple nodes manage and process the data for the subsequent dissemination of the results. In contrast, flat approaches operate on top of a random topology, such as meshes, which do not dictate the paths to collect and disseminate information. Instead, communication takes place between arbitrary nodes, mainly relying on concepts of gossiping [37, 14] to communicate. Due to the close relation between flat approaches and the applied communication pattern of gossiping flat approaches are often referred to as *gossip-based approaches*, which we use interchangeably throughout the thesis. The following description of applied monitoring topologies ranges from flat topologies over flat hierarchies to normal hierarchies. For the different types of topologies we present the identified concepts to create a topology and to identify communication partners. In addition, we mention if dedicated devices are required for the establishment of a separate topology.

3.1.1 *Flat Topologies*

Decentralized monitoring mechanisms that belong to this category avoid the creation of a dedicated topology for the data exchange. Instead, they gossip information between two arbitrarily selected nodes. For the arbitrary exchange of data they do not require any specific functionality from the communication layer below as long as the neighborhood can be identified. Due to the simple design flat approaches exist for both fixed and mobile networks, such as MANETs.

In MANETs a node communicates with its neighboring nodes, i.e., nodes within its communication range. Among these neighbors, current approaches rely on *random neighborhood selection* to communicate [50] or simply *broadcast* information [180, 41]. In fixed communication networks approaches rely on the neighborhood, which, for instance, is provided by an existing P2P overlay at the overlay layer (cf. Section 2.4.2.2) or by additional services, such as the Peer Sampling Service [68]. Since the availability of broadcasting cannot be assumed, flat decentralized monitoring mechanisms rely on random neighborhood selection to communicate [69, 170, 174].

3.1.2 *Flat Hierarchical Topologies*

A hierarchical topology is considered to be flat, for example, if the hierarchy consists of two or three levels. Approaches that rely on this type of topology, exploit the advantages of a hierarchy but keep the arising maintenance overhead small. In general, we differentiate between *overlays of clusters*, *connected trees*, and *trees of clusters*, which we describe below. Decentralized monitoring mechanisms, which operate on an overlay of clusters, are primarily designed for MANETs. To establish the hierarchy approaches create clusters, identify cluster heads, and establish an overlay among the heads for the communication between them. The applied cluster techniques cover proximity-based [97, 173] or k-means [119] clustering with the identification of the corresponding heads. To reduce the overhead for the creation of clusters a subset of approaches [137, 119] relies on fixed nodes as heads.

Other approaches for MANETs establish connected trees, where the root of a tree collects information from the nodes inside that tree. At the same time, the roots interact with each other to exchange the collected information. Existing solutions

solely rely on mobile nodes of the MANET [8] or include dedicated fixed nodes [142], which create a separate topology to collect and disseminate information. *Auxiliary functions*[1] are applied that rely on the geographical distance to a root [142] or on device characteristics [151] (e.g., a node's energy level) to determine the level of a node in the tree.

Finally, trees of clusters exist for both fixed and mobile communication networks. In case of MANETs, ANMP [21] assumes a single three-tiered tree with a single root and cluster heads as inner nodes of the tree. Grid Box Hierarchy [51] establishes a multi-tiered tree, where each node of the tree corresponds to a cluster. The approach avoids the identification of cluster heads and determines clusters by logically splitting the geographic network into disjoint areas. Astrolabe [176], Ganglia [114], and Storia [121] constitute corresponding solutions for fixed communication networks. Contrary to the presented approaches for MANETs, they require an existing hierarchy to establish the tree of clusters.

3.1.3 *Hierarchical Topologies*

Hierarchical topologies that avoid the creation of multiple hierarchical topologies but incorporate every node into one hierarchical topology, more precisely a tree, are mainly designed for fixed communication networks. GAP [29] and Willow [175] rely on a single tree and require information about the neighborhood to establish the tree. GAP uses a spanning tree by periodically applying the Breadth-First Search algorithm on the participating nodes. Willow requires a separate addressing scheme, more precisely, a separate ID space, to establish and maintain its tree. It assigns its own IDs out of that ID space to the nodes and, similar to Kademlia [116], relies on the XOR-distance between the assigned IDs to arrange the nodes in a tree. Similarly, Graffi [46] and Li et al. [100] rely on a single tree but require a specific type of an overlay below to create the tree. More precisely, both approaches assume the existence of a Distributed Hash Table (DHT), which offers the functionality to search for nodes based on provided IDs. As specified in [154] a DHT represents a specific class of a P2P overlay, which uses its own ID space to identify nodes as well as data objects. The corresponding IDs are used to (i) query nodes, (ii) retrieve data objects, and (iii) define how the corresponding lookup for an ID is routed through the overlay. Paired with the respective auxiliary function both approaches apply these functions to determine positions in the tree [46] or to identify possible parents [100]. The DHT is used to confirm the final position of the node in the tree and to determine an effective parent.

Other decentralized hierarchical monitoring mechanisms [182, 65, 183] solely rely on a DHT to create trees and avoid the application of an additional auxiliary function. Instead, every monitored attribute is mapped to an ID from the ID space of the DHT. Afterwards, the functionality from the DHT is used to determine responsible nodes based on the ID of an attribute. Subsequently, the attribute-related information is routed to the responsible node. The paths from all nodes to the responsible node for an attribute result in a tree, which is rooted at the responsible node. The remaining

1 In the context of the hierarchy creation an auxiliary function calculates the position of a node in the hierarchy. The function may use a node's context as input to generate a hierarchy-related ID, the immediate position, or the level for a node inside the hierarchy.

nodes become either inner nodes or leaves of that tree. This procedure is executed for every monitored attribute, which leads to a forest of trees. San Fermín [18] also creates a forest of trees, but in this case, the forest is used to create a robust approach instead of using one tree per attribute. Contrary to [182, 65, 183], San Fermín uses a DHT to identify nodes that share the same ID prefix of a given length.

In contrast to the previous approaches RAIC [105] constitutes an exception, because it is not designed for fixed networks but for MANETs. The crucial difference of RAIC to the approaches described above results from the fact that RAIC creates a tree on-demand, which lasts for a short amount of time. A query for the global view of attributes is spread in the network and the results are collected over the resulting tree at the querying node. The approach constitutes no viable solution for our application scenarios for two main reasons:

- Multiple nodes might create the temporary topology in parallel, leading to an unnecessary overhead.
- Only the root of the tree is provided with a global view of the network, while other nodes obtain a partial view.

3.1.4 *Conclusions*

As a result of the simple construction and minor maintenance overhead flat solutions appear to be a valid alternative for MANETs. However, a major disadvantage stems from the fact that the monitoring results are location-agnostic and always characterize the whole network. Due to the gossip-based communication paradigm paired with the mobility of nodes the current information at a node consists of data from nearby and distant nodes. Therefore, it is challenging to provide location-aware results, as demanded by the identified research challenges (cf. Section 1.2).

Approaches, which rely on a single tree or a forest of trees, do neither represent a viable solution for MANETs. They try to integrate each node into the hierarchy and are hard to maintain in dynamic networks. These trees of the surveyed decentralized monitoring mechanisms rely on parent-child relations, which must be frequently checked to identify failures. In our previous work for fixed communication networks [156, 158] we already revealed the negative impact of dynamic networks on hierarchical decentralized monitoring mechanisms. We showed that parent-child relations already suffer from an average online time of 15 min or 30 min of a node in the network.

Consequently, approaches with a flat hierarchical topology appear to be the monitoring topology of choice for two reasons. First, the hierarchy enables the provisioning of location-aware monitoring results. Lower levels provide detailed information with a local or regional scope, whereas higher levels summarize these scopes for a global view. Second, hierarchies are kept flat to reduce the negative influence of node mobility and limit the maintenance to a few layers.

3.2 DATA COLLECTION

This section details how the monitored attributes are collected from all participating nodes, using the established topology. As already stated in Section 2.4.2.3 the set

of monitored attributes may consist of diverse attributes from multiple layers of the considered communication network, covering attributes from the link layer up to the application layer. The process of data collection can be divided into four categories, which specify

- how a node measures the attributes;
- how the nodes communicate to collect the data;
- when the communication is triggered;
- how the nodes aggregate the data to reduce the size.

We describe the tasks of the first three categories in the remainder of this section, whereas data aggregation is separately detailed in Section 3.3. Data aggregation constitutes an important and frequently applied technique to process data, because it still reveals statistical insights into the network state, while the size of the data is compressed.

3.2.1 *Measurement Type*

The *measurement type* specifies if attributes are actively or passively measured. We classify the data collection as active if these attributes are locally measured by every node. Contrary, passive measurements are taken if a fraction or even dedicated nodes measure the required attributes by sniffing and overhearing packets from the remaining nodes [149]. The surveyed approaches reveal that all examples for fixed communication networks and the majority of MANETs rely on participating nodes and active measurement techniques to monitor the network. Mesh-Mon [119] and OBELIX [142] constitute exceptions, because they complement the active measurements by passively measuring the network. OBELIX even creates a separate topology on top of dedicated static nodes to collect and process the data.

3.2.2 *Data Exchange Strategy*

Once measured, the *data exchange strategy* specifies how the nodes interact with each other to collect the data. As outlined in our previous work [156, 158] we differentiate between *push-* and *pull-based* data exchange. In terms of a push-based data exchange a node always sends its monitored attributes to one or several nodes. If a pull-based data exchange is applied, a node requests data and subsequently receives data from one or several nodes. Based on the surveyed decentralized monitoring mechanisms, it is observable that the majority of approaches considers push-based data collection. Only two of the surveyed approaches, i.e., San Fermín [18] for fixed and RAIC [105] for mobile networks, exclusively rely on pull-based data collection.

Among the two extreme cases, especially hierarchical approaches are suitable to combine both strategies for two main reasons.

- The hierarchical topology specifies the paths to collect the data and limits the set of potential receivers, which enables the design of more focused and optimized collection strategies. In [100] nodes exploit the hierarchy to push a set

of commonly accepted attributes, whereas single nodes pull node-specific attributes. In ANMP [21] a node always pulls attributes but pushes them if the corresponding values exceed predefined thresholds. PRISM [65] initiates an isolated pull-based data collection, if it observes that the underlying tree has been recently restructured or got broken. SDIMS [182] and Shruti [183] choose between push- or pull-based data collection dependent on the characteristics of the monitored attributes. These characteristics are detailed in Section 3.2.4.

- Different hierarchy levels may be used to assign different collection strategies per level of the tree. In overlays of clusters or trees of clusters data are pushed inside a cluster and pulled along the tree [114, 173, 97], whereas connected trees, for instance, pull data at lower levels and push them at higher levels [151, 142].

Contrary to the flexible design of strategies for hierarchical approaches, gossip-based approaches usually push information to collect data. In general, a node randomly selects one or several neighbors and pushes the data [181, 174]. In MANETs, some approaches broadcast data to all neighbors in the communication range [180, 41]. *Push-pull-based* gossiping approaches [69, 50] only push the data to one randomly selected neighbor but receive and process the local data from the opponent in turn.

3.2.3 *Exchange Trigger Strategy*

The *exchange trigger strategy* is closely coupled with the data exchange strategy and specifies if and when the data exchange is triggered. In general, we differentiate between a *periodic* and an *event-driven* trigger. The periodic trigger defines intervals to constantly exchange data. Event-driven strategies only exchange data if an event was fired. Events are fired if the value of an attribute changes or if a node decides to push its own data to or pull data from one or several other nodes.

The surveyed approaches reveal that gossip-based solutions for MANETs [180, 50, 41] or fixed networks [81, 170, 139] utilize periodic triggers to exchange data. While these approaches do not impose strict rules on the monitoring topology and potential communication partners, they require this fixed interval, also denoted as *cycle*, to organize common tasks, including the periodic data collection.

Among the hierarchical approaches, several solutions [51, 151, 8] rely on a static periodic trigger. Other solutions try to relax the static behavior and allow for adapting an interval based on a given criterion. In Astrolabe [176] the interval can be independently set for each level. In [46] the interval is fixed, but the root regularly synchronizes the update intervals in the hierarchy so that data are immediately passed from leaves towards the root. Although PRISM [65] uses a periodic update strategy, not every interval is used to push data towards the root of the tree. Instead, a node forwards an attribute (i) if the related value significantly changed or (ii) if the interval from the previous to the next update exceeds a given threshold. Other approaches solely rely on an event-driven data exchange, which is either triggered by nodes [29, 105, 18] or if the values of attributes change [175, 182, 183].

In between, Li et al. [100] and Chen et al. [21] offer both strategies simultaneously. They use periodic triggers to collect common attributes, but offer the possibility to trigger the collection of node-specific attributes on-demand or if values of attributes exceed predefined thresholds. Other approaches utilize a *node-dependent*

strategy, where the trigger depends on the current context of a node, e.g., its hierarchy level. For instance, Ganglia [114] and Storia [121] use event-driven data collection in clusters but periodically collect data along the tree, whereas Tuduce and Gross [173] implement it vice versa.

3.2.4 *Conclusions*

Based on the surveyed concepts for data collection several conclusions are drawn. In terms of the measurement type it is obvious that passive techniques do not represent a viable alternative for us, since the corresponding approaches rely on dedicated fixed nodes. As stated within the description of the scenario we exclude fixed nodes in our thesis and just focus on mobile nodes. Due to the close relation of data collection and trigger strategies we combine our conclusions for both categories.

The periodic transmission of data, either pull- or push-based, is mainly useful, when fluctuating attributes are monitored. Otherwise, the constant transmission results in an unnecessary overhead, if information does not change between two or several consecutive transmissions. In contrast, the periodic communication may be used to piggyback additional information about the topology to maintain a valid structure and react on node or link failures [100]. PRISM [65] avoids the collection of useless data and communicates during an interval if an attribute significantly changed or sufficient time elapsed (cf. Section 3.2.2). However, PRISM relies on a hierarchy and demands that parent-child relations are valid for a longer period of time so that significant changes of an attribute or the elapsed time are tracked by the child and corresponding parent. The basic assumption does not hold in MANETs, because connections between nodes in parent-child relations mostly exist for some seconds.

A decentralized monitoring mechanism, solely relying on an event-driven collection strategy, may reduce the resulting communication overhead, because data collection is only executed if an attribute changes or a node requires recent information. But as stated by Dam and Stadler [29], the reduced traffic leads to an increased instability of the hierarchy, requiring that the underlying topology is managed based on separately transmitted information.

If events are triggered by changing attributes, monitoring fluctuating attributes leads to a considerable communication overhead, since each change leads to an update affecting parts or the whole topology. To prevent this overhead SDIMS [182] and Shruti [183] rely on an event-based trigger strategy combined with an attribute-dependent collection strategy. A rather constant attribute is pushed up the tree, whereas fluctuating attributes trigger only a local update at the node itself and are pulled if required. Similar to PRISM, SDIMS and Shruti require a rather stable network, thus, are inappropriate for MANETs.

3.3 DATA AGGREGATION

So far, we just discussed how and when nodes communicate to collect data. In this section we detail how a node aggregates the data from other nodes and how it integrates its locally measured data.

3.3.1 *Hierarchical Computation Property*

The *hierarchical computation property* was introduced by Madden et al. [109] and defines how attributes are aggregated by a hierarchical monitoring mechanism. Following the nomenclature from Yalagandula and Dahlin [182], a common aggregation function for an hierarchical approach is defined as

$$V_{i,type,name} = f_{type}\left(V^0_{i-1,type,name}, V^1_{i-1,type,name}, \ldots, V^{k-1}_{i-1,type,name}\right) \quad (1)$$

In this equation i is the current level of the node, which aggregates the results from its k sub-trees V^x, which are rooted at level $i-1$. The function f_{type} specifies how an attribute of *type* and *name* is aggregated. Dependent on the decentralized monitoring mechanism, attributes of a certain type may exclusively be assigned to one aggregation function f_{type} so that the type of an attribute and the aggregation function match. For instance, an attribute that represents the *number of nodes* in the network or the *number of requests* for a certain data item may be assigned to the aggregation function f_{count} that counts the measured values of both attributes. Other approaches may avoid a specific assignment, but define a set of aggregation functions, which are applied on every monitored attribute. For example, Graffi et al. [47] define seven aggregation functions so that a processed attribute is represented by seven key figures after the application of the different aggregation functions. In our previous work [156, 158] as well as throughout the thesis we rely on the second of the discussed strategies and define a set of aggregation functions to process every monitored attribute.

In general, an aggregation function f is said to satisfy the hierarchical computation property if (2) holds. We rely on [182] to explain this property based on the example of the average aggregation function f_{avg}, which constitutes one of many ways to aggregate data. If a node averages the received values using $f_{avg} = \frac{1}{k} * \sum_{i=0}^{k-1} V^i$ for its $|V^x| = k$ sub-trees, the function does not satisfy the hierarchical computation property. Instead, if nodes calculate the two aggregates f_{sum} and f_{count} that both satisfy the hierarchical computation property and where f_{count} represents the number of included attributes, f_{avg} is calculated by $f_{avg} = f_{sum}/f_{count}$, preserving the hierarchical computation property.

$$f\left(V^1, \ldots, V^n\right) = f\left(f\left(V^1, \ldots, V^{s_1}\right), f\left(V^{s_1+1}, \ldots, V^{s_2}\right), \ldots, f\left(V^{s_k+1}, \ldots, V^n\right)\right) \quad (2)$$

The major advantage of this property results from the fact that each node may already aggregate partial results from its children before forwarding the data. Moreover, nodes must not necessarily inject their local measurements at the lowest hierarchy level, but are able to integrate their measurements at the current level. If the hierarchy considers geographical positions of nodes and preserves their locality by arranging nearby nodes next to each other, the hierarchical computation property preserves this locality during aggregation: aggregation at higher levels summarizes results from distant places, whereas lower levels dispose of monitoring information from nearby places.

3.3.2 *Mass Conservation*

Flat approaches avoid the creation of a hierarchy with dedicated relations between nodes or clusters and rely on gossiping [37, 14] so that arbitrary nodes communicate

with each other. To apply different aggregation functions to monitored attributes concepts from hierarchical approaches are hardly applicable. For that reason gossip-based approaches, which belong to this class of decentralized monitoring mechanisms, rely on concepts of *mass conservation* [81] and exploit gossiping to aggregate the monitored data [181].

As mentioned in Section 3.2.2, gossip-based solutions are divided into push-based approaches, which are mainly based on Push-Sum [81], and push-pull-based approaches, which adopt Pairwise-Avg [69]. Both algorithms are cycle-based, meaning that each node executes the algorithm once per cycle. The time-based synchronization is reflected by the selected exchange trigger strategy, since all approaches rely on a static periodic data exchange (cf. Section 3.2.3).

In the following we briefly sketch the concept of mass conservation, while we refer the interested reader to [81, 69, 124, 181] for further details on mass conservation. Every node K maintains two state variables $s_{t,K}$ and $w_{t,K}$, where s holds the monitored value of an attribute, w corresponds to the weight of a node, which is set to one in the beginning, and t denotes the current cycle of the node. During a cycle, a node sends its state variables $s_{t,K}$ and $w_{t,K}$ to one or multiple receivers. In turn, it may receive both state variables from other nodes. The received state variables are combined with the own local state variables so that the two invariants (3) and (4) of mass conservation hold. The first invariant implies that the sum over all attributes does not change over time and is equal to the sum of the initial values x_K over all nodes. The second invariant demands that the sum over all weights corresponds to the number of nodes in the network. If both invariants hold, then $s_{t,K}/w_{t,K}$ converges to the effective average of the monitored attribute at every node K.

$$\sum_K s_{t,K} = \sum_K x_K, \quad \forall t \geq 0 \tag{3}$$

$$\sum_K w_{t,K} = n, \quad \forall t \geq 0 \tag{4}$$

Several techniques exist to incorporate new values of the monitored attributes and to obtain a recent view of the system. In LPS [41] nodes start the gossip-based aggregation if they require a recent view, which we denote as *node-initiated mass conservation*. A node stops the execution, if it observes that the local aggregates do not change anymore. *Epoch-synced mass conservation* is applied, if data collection and result dissemination are periodically restarted to integrate fresh values. Jelasity et al. [69] introduce the concept of *epochs* and restart the protocol if an epoch, consisting of a fixed number of cycles, has elapsed. Terpstra et al. [170] periodically start a new epoch based on the same criterion as LPS [41]. Finally, the native Push-Sum approach [81] and G-Gap [181] enable the continuous integration of new values without a restart, which we denote as *continuous mass conservation*. In this case, each node inserts the difference between the new and old value of a monitored attribute.

3.3.3 *Population-Based Aggregation*

The concepts of population algorithms [5] represent another procedure to aggregate data. In general, population protocols consider the participating nodes as finite state machines with limited memory to store information. Similar to mass conservation-

based approaches population protocols are cycle-based. During a cycle, pairs of nodes interact with each other to carry out the required computation.

On the basis of population protocols, the collection and immediate aggregation of monitoring data can be implemented. Guerrieri et al. [50] introduce the Two-Phases algorithm for mobile networks, where two neighboring nodes interact with each other during one cycle. During the interaction, one node collects and aggregates the information, which is repeated until one node obtains the global view of an attribute, incorporating data from all nodes.

Gossipico [174] constitutes a similar approach, which targets fixed communication networks. During the interaction, a node sends its data, if available, to another node, which collects the data. If a node does not have data, it transmits its current view of monitored attributes to the other node. To speed up the collection leader election is used to head the collection of data towards the leader. Gossipico is again detailed in Chapter 5 to highlight relevant ideas for the development of our decentralized monitoring mechanisms.

Similar to mass conservation-based approaches population algorithms avoid the creation of a hierarchy and rely on the gossip-based communication paradigm to let nodes exchange information. However, as indicated by Wuhib et al. [181], there is a subtle but significant difference between both classes of approaches, because population protocols use *gossiping to communicate*, whereas mass conservation-based approaches us *gossiping to aggregate*.

3.3.4 *Conclusions*

The hierarchical computation property enables flexible aggregation within a hierarchy and provides location-aware results if the establishment of the underlying hierarchy is location-aware. However, the property requires a hierarchy, which, in turn, must be established and maintained. Moreover, the calculation of some aggregates (e.g. counts or sums) is highly duplicate sensitive: if the tree is restructured or nodes send updates too often, aggregates are calculated based on duplicate data.

In contrast, mass conservation-based aggregation handles duplicate contributions from one or several nodes as long as the mass of the system does not change. However, the integration of recently measured values is more complicated than for tree-based approaches. Either, the aggregation must be periodically restarted, requiring that all participating nodes are synchronized to some degree [69, 170], or, if continuous mass conservation is utilized, each node must store additional state information to restore the correct mass in the presence of failures. In fact, mass conservation assumes that no information gets lost and the number of nodes is fixed [81]. To relax these assumptions, periodic restarts must be triggered or additional calculations must be executed to counteract the changing mass of a system.

Finally, population protocols inherit the flat structure and gossip-based communication pattern from mass conservation-based approaches but just gossip to communicate. Based on the sketched procedure of population protocols, the performance, i.e., the collection time, heavily depends on the interaction pattern of nodes. For instance, data collection is delayed if the interaction primarily takes place between clusters of nodes [50]. Solutions to accelerate the collection exist [174] but may exhibit similar characteristics as hierarchical approaches: the accuracy of the monitoring mechanism

drastically decreases if data or nodes next to the highest hierarchy level are lost or not accessible.

3.4 RESULT DISSEMINATION

The *result dissemination* of a decentralized monitoring mechanism defines how interested nodes receive the results: if nodes must request the results, the monitoring mechanism applies a *reactive* result dissemination. If nodes receive results without a request, the results are *proactively* disseminated.

Due to push-based data collection gossip-based approaches, such as [170, 41, 50, 174], immediately disseminate the results in a proactive manner among all nodes. In tree-based solutions reactive and proactive result dissemination strategies are applicable. The majority of surveyed approaches for mobile networks exclusively relies on one of both strategies and either proactively disseminates the results over the established topology [175, 46] or just reacts on requests from nodes [105, 65, 97]. For fixed networks a smaller fraction of decentralized monitoring mechanisms restricts itself to one of the two strategies. In between, approaches exist that provide both strategies dependent on the characteristics of the monitored attributes. For instance, Li et al. [100] differentiate between commonly accepted or node-related attributes, whereas SDIMS [182] and Shruti [183] proactively disseminate constant attributes, while fluctuating attributes are reactively fetched.

3.4.1 *Conclusions*

Apart from gossip-based approaches, it is observed that solutions for fixed communication networks are mainly dominated by a proactive result dissemination, whereas solutions for MANETs prefer reactive strategies. If the monitored information is not necessarily required by the participating nodes, reactive approaches are beneficial, because results are only transmitted if required. However, based on the made assumptions of this thesis, the participating nodes require the information so that they obtain a current view of the communication network to decide if the communication network must be adapted (cf. Section 2.4.2.3). As pointed out in our previous work for decentralized monitoring in MANETs [157] the collection of monitoring results at a set of nodes, e.g. the cluster heads [97] or roots of connected trees [8], leads to several drawbacks. The availability of monitoring results directly relates to the presence and accessibility of those nodes and a fraction of nodes is in charge of serving the requests from a whole network.

3.5 SUMMARY AND CONCLUSION

In this chapter we surveyed the related work on decentralized monitoring mechanisms for fixed communication networks and MANETs and examined the utilized concepts of existing approaches to implement the three basic components of a decentralized monitoring mechanism. Based on the overview, we refined our classification of the three main components and identified related categories to provide a more detailed classification of the utilized concepts for mobile and fixed networks.

Category	Explanation
Topology	Specifies the topology of a decentralized monitoring mechanism
Topology construction	Specifies how the respective topology is created
Data exchange strategy	Specifies how a decentralized monitoring mechanism exchanges data
Exchange trigger strategy	Specifies when a decentralized monitoring mechanism triggers the data exchange
Aggregation type	Specifies how a node of a decentralized monitoring mechanism processes data

Table 4: Overview on the identified categories of a decentralized monitoring mechanism

Figure 9 and 10 depict the three basic components with the associated categories and present the utilized and implemented concepts of an approach. For the sake of a clear overview we restrict ourselves to the most relevant categories and divide the set of approaches into two subsets, which are separately illustrated. Figure 9 shows the solutions that rely on trees, trees of clusters, forest of trees, or connected trees, whereas Figure 10 depicts the rather flat approaches, comprising meshes and overlay of meshes[2]. The approaches with a rectangular marker represent solutions for fixed communication networks, whereas the circular markers represent solutions for MANETs. Table 4 briefly summarizes the meaning of every depicted category.

Based on the survey and classification of existing approaches for decentralized monitoring mechanisms, we identified generally accepted concepts and best practices. They serve as basis for the design and development of our approaches and are briefly summarized in the following. With respect to the underlying monitoring topology the survey outlines that decentralized monitoring mechanisms for MANETs rely on a rather flat structure. Contrary to approaches for fixed networks that build single or even multiple trees, incorporating all nodes, monitoring hierarchies for MANETs only consist of a few levels. The flat hierarchy does not necessarily cover the whole MANET, but only a fraction. In this context the survey reveals that existing approaches connect the cluster heads [173, 97] or roots from different trees [21, 8] with each other to extend the monitoring scope.

Focusing on the collection of data and dissemination of results, the applied strategies unveil that the majority of decentralized monitoring mechanisms for MANETs relies on static strategies. This covers the strategies to trigger the data exchange as well as the strategies for the data exchange itself. In contrast, decentralized monitoring mechanisms for fixed networks exploit the rather static nature of the communication network and define adaptive strategies, which are particularly designed for hierarchical approaches. The adaptive strategies switch between a push- and pull-based data exchange [182, 65]. In addition, they apply adaptive periodic intervals [176, 46] to trigger the data exchange or even switch between an event-based

2 In both figures some terms are abbreviated to fit the corresponding figure. The abbreviation HCP stands for hierarchical computation property, RNS denotes random neighborhood selection, and MC stands for mass conservation.

and periodic trigger strategy [100]. Similar strategies are observable for the dissemination of monitoring results. Contrary to the flexible strategies for fixed networks, the survey underpins that the majority of solutions for MANET permanently sticks to one strategy to exchange data as well as to trigger the exchange.

More important than the identification of generally accepted concepts and best practices are the identified shortcomings and problems that have to be tackled or avoided. In the following we (i) outline some aspects, which differ from existing monitoring solutions and (ii) detail where we go beyond the state of the art. In general, we abstain from an additional and separate infrastructure with dedicated devices to monitor a MANET as in [137, 142]. The developed decentralized monitoring mechanisms integrate every participating node into the monitoring process, including (i) the establishment and maintenance of the topology, (ii) data collection, and (iii) result dissemination. Our approaches are not designed to provide monitoring results for a third party, for instance, a network operator, to analyze the network state and identify countermeasures [119, 142]. Instead, the nodes themselves are responsible to (i) analyze the results, (ii) identify appropriate countermeasures, and (iii) adapt the network. Consequently, we go beyond existing monitoring solutions for MANETs and adapt concepts from approaches for fixed networks [176, 183, 46] to ease the access of monitoring results. As a consequence, insights on the network state are not limited to a subset of nodes [137, 97, 8, 142], which must be queried. Instead, monitoring results are disseminated in the network to provide each node with the information. Moreover, the developed decentralized monitoring mechanisms do not depend on a generic and monitoring-agnostic routing protocol, but implement their own routing strategies to enable data exchange even if the underlying routing protocol fails [119].

In addition to the extensions and new concepts that address a decentralized monitoring mechanism for MANETs we present specific aspects to extend hierarchical and flat approaches. Regarding the establishment of a hierarchical topology, we design a hierarchy that does not rely on parent-child relations between two nodes, as proposed by other solutions [100, 65, 18, 46]. Our previous results for decentralized monitoring mechanisms in fixed communication networks have shown that the performance of a decentralized monitoring mechanism with a hierarchical topology degrades in highly dynamic networks [156, 158]. Consequently and as already proposed by other approaches [21, 51, 121], we rely on parent-child relations between areas, which consist of multiple nodes, to handle the inherent dynamics of MANETs. In terms of flat and gossip-based monitoring mechanisms we promote the development of those approaches to extend the small number of existing solutions for MANETs. In fact, we go beyond existing approaches that (i) are restricted to static wireless or delay tolerant networks or (ii) operate under simplified assumptions. With the design of our flat and gossip-based approach we exploit the robust characteristics of gossiping without the need for a hierarchy. Contrary to the majority of flat approaches, which gossip to aggregate and rely on the concept of mass conservation [170, 181, 50, 41, 139], we use gossiping to exchange information. Based on this design decision, we leverage the robust communication pattern but abstain from the concepts of mass conservation to avoid mass loss.

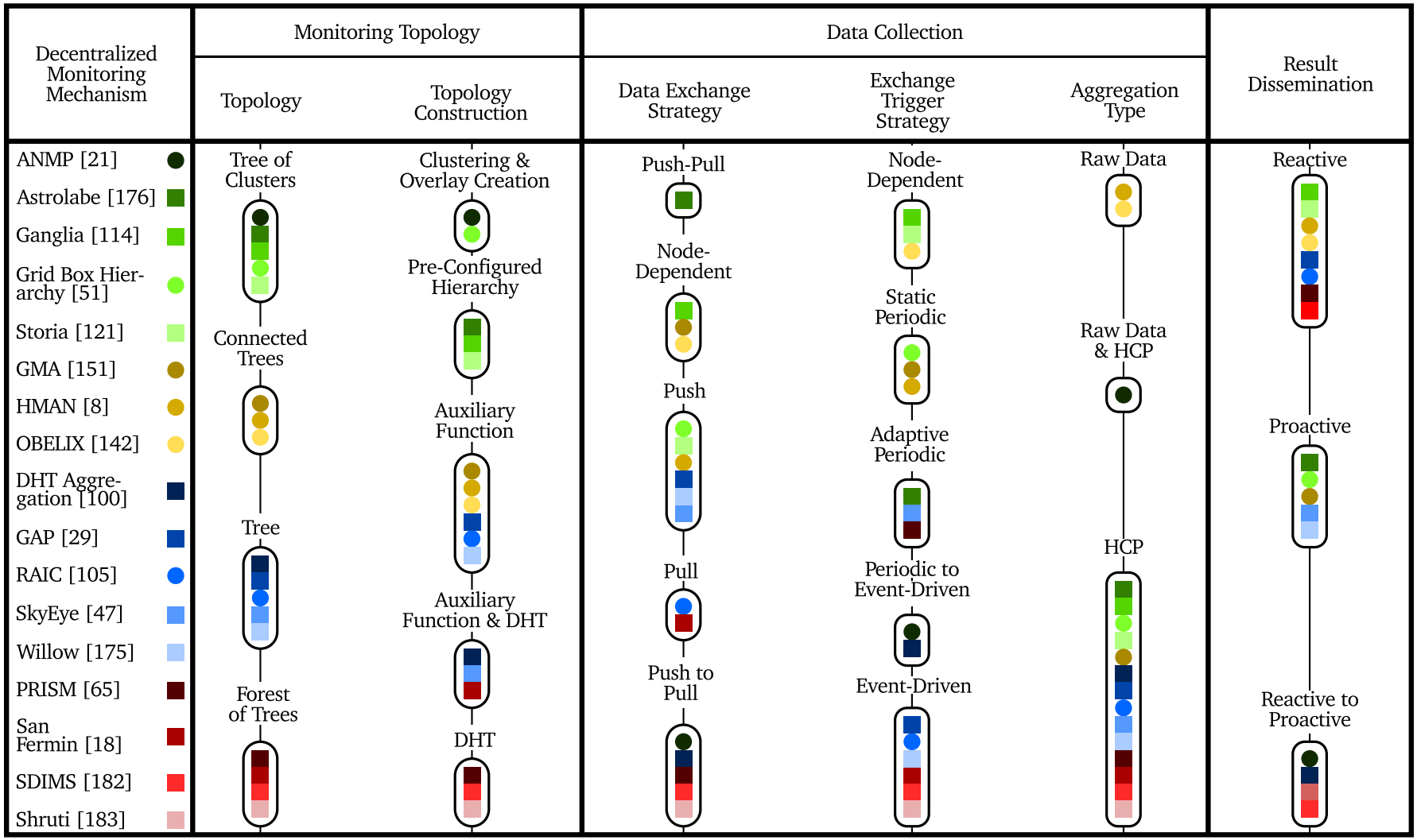


Figure 9: Overview and classification of decentralized monitoring mechanisms, which rely on trees, trees of clusters, forests of trees, and connected trees as topology. The approaches with a rectangular marker represent solutions for fixed communication networks, whereas the circular markers represent solutions for MANETs.

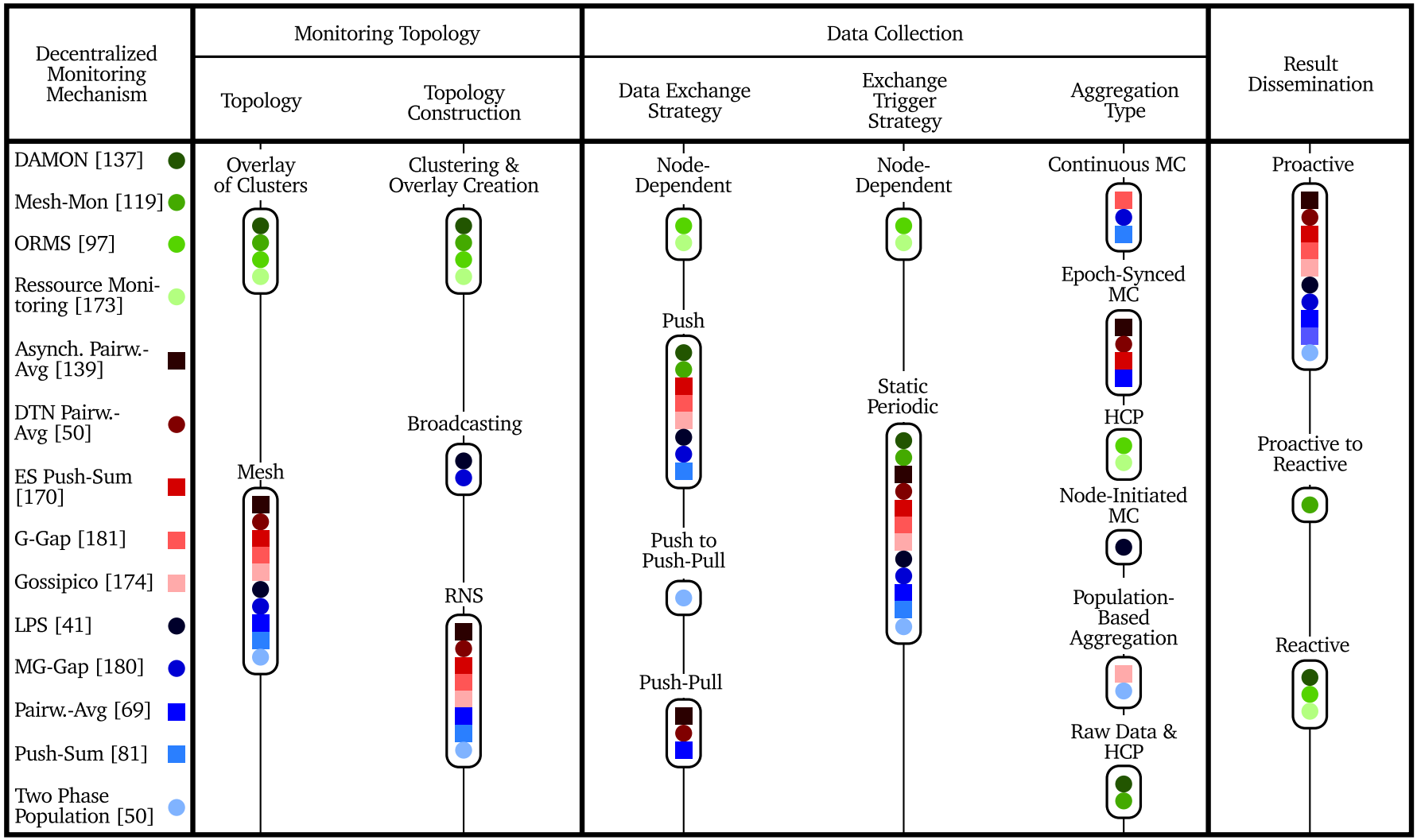


Figure 10: Overview and classification of decentralized monitoring mechanisms, which rely on meshes and overlays of meshes to create a topology. The approaches with a rectangular marker represent solutions for fixed communication networks, whereas the circular markers represent solutions for MANETs.

4

BLOCKTREE.KOM: HIERARCHICAL AND LOCATION-AWARE MONITORING

Given the principles of decentralized monitoring as well as an overview about generally accepted and applied concepts for decentralized monitoring mechanisms, this chapter details our solution, named BlockTree.KOM. BlockTree.KOM is a decentralized monitoring mechanism targeted at MANETs. It monitors and provides information about the current state of the network and the participating nodes. This information serves as the basis for the adaptation of a communication network, as described in Section 2.2. In transition-enabled communication networks, as envisaged by MAKI [49], the monitored information is utilized to plan and execute transitions or multi-mechanisms adaptations.

BlockTree.KOM operates on top of a hierarchical topology to provide location-aware monitoring results. Similar to the majority of the surveyed decentralized monitoring mechanisms for both fixed and mobile networks BlockTree.KOM integrates all covered nodes into the process of monitoring. Consequently, the nodes are in charge to create the hierarchy and to monitor the set of attributes. The established hierarchy serves as a basis to identify the responsible nodes for data collection and result dissemination. BlockTree.KOM strictly limits its scope on a certain spatial area to (i) avoid the transmission of information from distant nodes, (ii) save the resources of participating nodes, and (iii) reduce the needless utilization of the communication medium. As a result and similar to other approaches [51, 151] the underlying hierarchy of BlockTree.KOM is configurable and may be limited to a maximum height. Therefore, the monitored area of BlockTree.KOM either covers the whole MANET or separate instances are put together to cover the entire MANET, as discussed in Section 2.4.2.3.

As depicted in Figure 11 BlockTree.KOM encompasses three components for its proper operation, which are represented by the blue boxes.

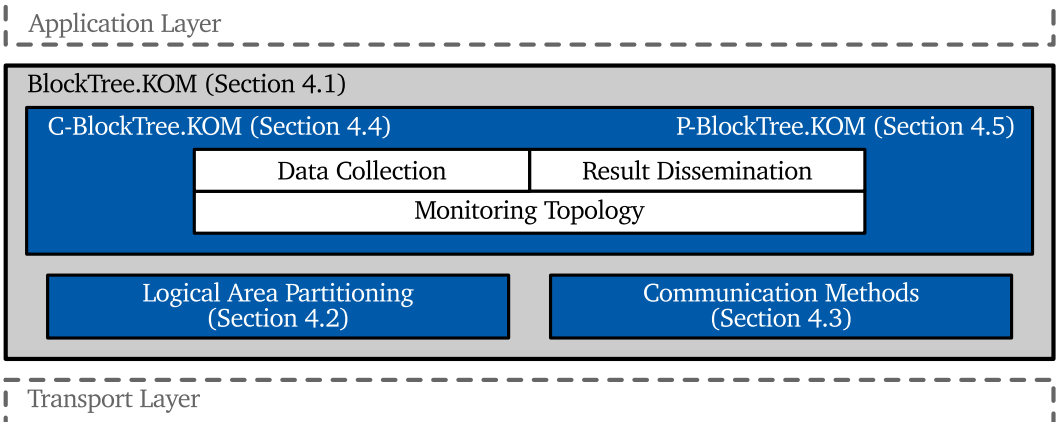


Figure 11: Overview on BlockTree.KOM

DECENTRALIZED MONITORING MECHANISM. BLOCKTREE.KOM is represented by P-BLOCKTREE.KOM and C-BLOCKTREE.KOM, which constitute the actual mechanisms to monitor a MANET and consist of the three well-known components to establish the monitoring topology, collect the data, and disseminate results. Both approaches rely on BLOCKTREE.KOM's concepts (cf. Section 4.1) but implement them differently, as discussed in Section 4.4 and 4.5, respectively.

LOGICAL AREA PARTITIONING. This component provides the functionality for the logical partitioning of an area into *blocks*. These blocks build the basis to create the hierarchical topology and serve for the name of our approach. Relying on our classification of different monitoring topologies (cf. Section 3.1.2), BLOCKTREE.KOM's hierarchical topology may be classified as a *tree of clusters*, where the clusters correspond to the obtained blocks.

COMMUNICATION METHODS. BLOCKTREE.KOM does not require an existing routing protocol from the communication layers below to communicate and exchange information. Instead, it uses the simple yet robust methods of this component, which are tailored to the requirements for the data exchange in BLOCKTREE.KOM. Consequently, BLOCKTREE.KOM must not operate on service-agnostic routing protocols but uses its own routing methods.

In the following and as depicted in Figure 11 we start with a conceptual overview on BLOCKTREE.KOM, presenting its fundamental concepts, while facing the identified challenges (cf. Section 1.2). Subsequently, we detail how the network is partitioned and describe the routing methods in Section 4.2 and 4.3, respectively. Given the basic concepts of BLOCKTREE.KOM as well as the possibility to partition the network and to communicate, we detail the actual implementation of BLOCKTREE.KOM. As highlighted by the blue boxes in Figure 11 we provide two different approaches that implement the identified concepts of BLOCKTREE.KOM in two different ways. The concentrating approach, denoted as C-BLOCKTREE.KOM, is presented in Section 4.4, while Section 4.5 deals with the planar approach, referred to as P-BLOCKTREE.KOM.

4.1 CONCEPTUAL OVERVIEW

This section provides insights about the conceptual design of BLOCKTREE.KOM and discusses the derived concepts to monitor MANETs. First, we present the basic ideas to provide the demanded location-aware results on top of a hierarchical topology that withstands the dynamic nature of MANETs and an error-prone communication medium. Given the topology, we describe the concepts for the communication between nodes to collect and disseminate information.

4.1.1 *Topology Concepts*

BLOCKTREE.KOM establishes a location-preserving hierarchical topology paired with data aggregation that satisfies the hierarchical computation property for the provisioning of location-aware results. To minimize the negative influence of mobility and the error-prone communication medium on the hierarchical topology BLOCKTREE.KOM introduces two fundamental concepts for the establishment of the hierarchical topology.

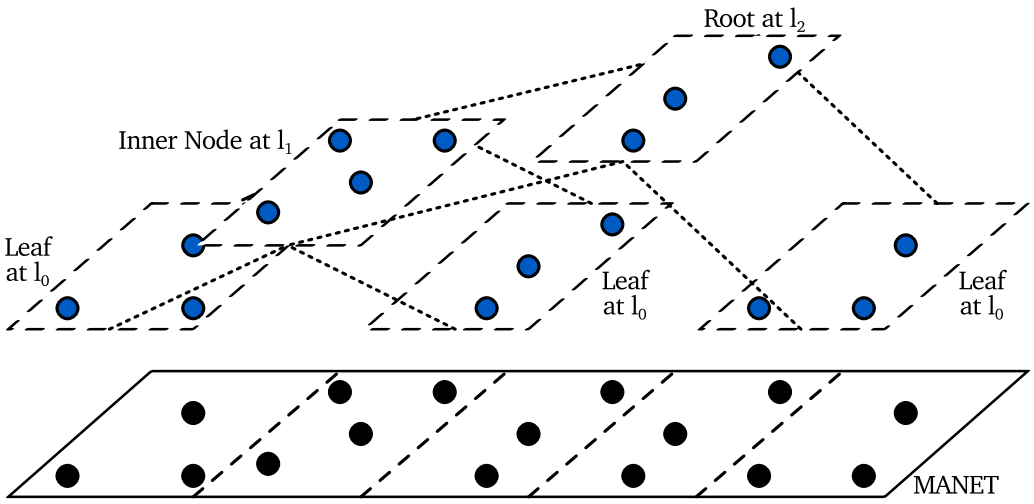


Figure 12: Example of a tree of clusters, more precisely, of blocks as monitoring hierarchy. The clusters correspond to blocks, which are obtained by a logical partitioning of the area.

TOPOLOGY CONCEPT ONE. BLOCKTREE.KOM does not rely on parent-child relations between two single nodes to create a hierarchy. Instead, the corresponding relation consists of multiple nodes to ensure that a disappearing node does not eliminate the relation or an arriving node replaces a current node in that relation.

TOPOLOGY CONCEPT TWO. BLOCKTREE.KOM reduces the required state information about the hierarchy to a minimum. A node does not maintain its relation to a parent so that the arrival or departure of nodes do not lead to stale or incomplete information.

Following these two concepts, BLOCKTREE.KOM operates on top of a tree of clusters, where the corresponding parent-child relations consist of multiple nodes to conform to the first topology concept. For the identification of clusters and members BLOCKTREE.KOM logically divides the area into blocks. Consequently, we denote the resulting monitoring topology as *tree of blocks* for the remainder of this thesis. A block with the covered nodes represents a leaf, inner node or root in the tree, as depicted in Figure 12. Section 4.2 describes the logical partitioning of an area into blocks in more detail. To complete the creation of the tree of blocks the blocks are hierarchically arranged, as indicated in Figure 12. The hierarchy levels are abbreviated as l_x, where x represents the respective level. Every block at any level l_x, except at the lowest level l_0, may have multiple child blocks and one parent block. The maximum level, which the established hierarchy may reach but not exceed, is given and configured by the system parameter l_{max}. Depending on the configuration of l_{max} and the spatial network size, the resulting hierarchy may cover the whole MANET or multiple instances must be run simultaneously to cover the entire MANET.

For the identification of a parent block nodes exclusively rely on their geographical position. Based on geographical positions, they are able to identify their current block, which serves in turn to determine the parent block. As a result no information about residing nodes in a block is required, thus, being in line with the second topol-

ogy concept. The determination of a parent block solely depends on the concrete implementation of BlockTree.KOM's component for the monitoring topology and is detailed in Section 4.4.1 and 4.5.1 for C- and P-BlockTree.KOM, respectively.

4.1.2 *Communication Concepts*

Subsequent to the description of the design of the hierarchical topology, we outline how nodes communicate and address other nodes in different blocks. Again, we provide an overview of the underlying concepts, on which the communication is based on and which are designed to handle the challenges that arise from MANETs and the wireless communication medium.

COMMUNICATION CONCEPT ONE. BlockTree.KOM exploits its cluster-based structure so that data are never exchanged between a single pair of nodes. Instead, the communication always happens between two sets of nodes to decrease the probability of data loss due to node mobility or churn.

COMMUNICATION CONCEPT TWO. Similar to the second topology concept BlockTree.KOM reduces the required state information at a node to decide how to handle incoming messages as well as where to send outgoing messages. So, nodes do not require additional data from neighboring nodes to identify valid senders or receivers. Consequently, the constantly changing neighbors of a node do not influence the transmission of information.

To comply with the first concept BlockTree.KOM relies on data-transmission oriented geocasting from the area of position-based routing protocols, as introduced in Section 2.3.3.2. Based on these routing protocols, we exploit the information of geographical positions to determine the route from a source to a destination. Similar to other geocasting protocols (cf. Section 2.3.3.2) the transmitted information is always intended for a set of nodes $\mathcal{D}$, which currently reside in the targeted geocast region. In this context $\mathcal{D}$ constitutes the set of addressed nodes in the geocast region. Consequently, we do not require a location service to determine the current geographical position of one or several nodes but exploit the knowledge about the location of blocks to determine the set of receiving nodes. To forward information to a geocast region data-transmission oriented geocasting protocols incorporate all intermediate nodes, which currently reside in the forwarding region so that multiple and redundant paths are established (cf. Figure 4). Finally, if a set of sources starts to geocast, the whole transmission of data always relies on multiple nodes to send, forward, and receive the data.

Storing (i) the position of the source and (ii) the region of the destination in a message, a receiving node simply relies on its current geographical position to decide to forward or process the message independent of the previous or next hop and type of the message. Furthermore, if properties of the forwarding region are defined, a receiving node may even drop the message, if the message has left the forwarding region. In Section 4.3 we describe the developed routing methods of BlockTree.KOM, which are based on the presented concepts and adopt ideas from data-transmission oriented geocasting.

4.2 LOGICAL AREA PARTITIONING

In this section we detail how BLOCKTREE.KOM logically divides the area into *blocks*, which serve as basis for BLOCKTREE.KOM to establish its hierarchical topology and to monitor MANETs. BLOCKTREE.KOM does neither rely on a central instance nor coordinator, which specifies how the considered area is divided. Instead, each node is capable to reproduce the logical splitting and to uniquely determine individual blocks, such as its own, parent, or child blocks. For the identification of blocks (i) nodes require a point of reference, for instance, the intersection of the equator with the prime median, as stated in our previous work [157], and (ii) they must determine their current geographical position. Since BLOCKTREE.KOM does not operate on *geographic coordinates* but *Cartesian coordinates*, we start with a description of the coordinate transformation, as detailed in Section 4.2.1. Relying on a Cartesian coordinate system, the corresponding nomenclature is introduced, which serves throughout the thesis to describe how BLOCKTREE.KOM operates in this coordinate system. Given the fundamentals, Section 4.2.2 deals with the logical area partitioning and the identification of blocks.

4.2.1 *Coordinate Transformation and Representation*

BLOCKTREE.KOM relies on a Cartesian coordinate system, because it eases the required calculations of distances between nodes or calculations of certain areas. For this reason the geographic coordinate system must be projected onto the Cartesian coordinate system. Geographic coordinates are specified by longitude and latitude and, for instance, may be provided by a GPS receiver. The system parameter $t_{\text{reqPosInterval}}$ of BLOCKTREE.KOM specifies after which amount of time a node retrieves and refreshes the information about its geographical position utilizing, for instance, its GPS receiver. Out of the different approaches for the projection of geographic coordinates on a plane [152], the *Universal Transverse Mercator (UTM) projection* is used, because it represents a frequently used projection for the creation of maps [152, 76]. The UTM projection wraps a cylinder around the earth, where the intersection of the cylinder and earth is denoted as *central meridian*. A map along the central meridian perfectly scales and nearby regions are projected onto that map with low distortion still being conformal [152]. To avoid an increasing error of the projection of distant places on a map the *UTM system*, which exploits the UTM projection, divides the earth into 60 *longitude zones* along the equator with a width of six degree in longitude and into 20 *latitude zones* along the meridians with a width of eight degree in latitude. The coordinates of a point on the projected map are expressed in meters and determined dependent on the associated meridian for that zone and on the hemisphere. *Easting* represents the distance from the associated median, while *northing* is the distance from the equator. For example, the geographic and UTM coordinates of the Multimedia Communications Lab in Darmstadt, Germany, are listed in Table 5. For further details on UTM or other projections we refer the reader to [152, 76].

After the transformation, geographic coordinates are expressed in a Cartesian coordinate system. To represent positions and execute calculations in that coordinate system we stick to the definitions and nomenclature for analytic geometry, as defined in Bronstein et al. [16]. In this thesis we consider two-dimensional planar coordinate

Type	Fields	Values
Geographic coordinates	(lat, lon)	(49.874774°, 8.660512°)
UTM coordinates	(longitude-zone, latitude-zone, easting, northing)	(32, U, 475.6068 km, 5524.7629 km)

Table 5: Example for the geographic and UTM coordinates of the Multimedia Communications Lab, TU Darmstadt, Germany

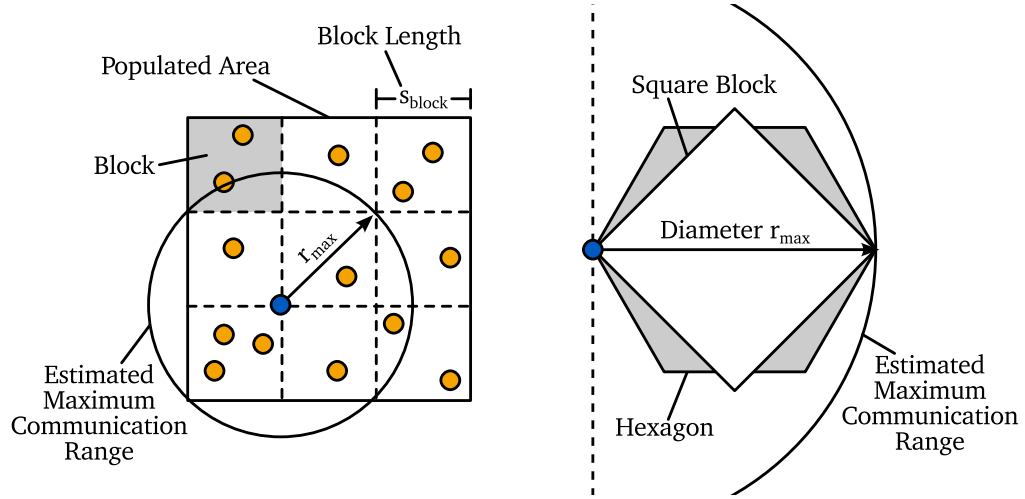
systems, because we assume rather flat areas and regions for the deployment of MANETs, where the impact of height, representing the third dimension, is negligible. The axes of the planar Cartesian coordinate system are represented by the two orthogonal unit vectors

$$\vec{e_1} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \vec{e_2} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

During the thesis, we will use the subscripts 1 and 2 to refer to the x- and y-coordinates of a point or a vector. To avoid ambiguity of subscripts, which are, for instance, used to enumerate intermediate nodes or nodes in the geocast region (cf. Section 2.3.1), we access x- and y-coordinates of a point or vector using a numerical index, e.g., $A[1]$ or $\vec{a}[2]$. In the coordinate system points are represented by capital letters, such as A and B, where $\overrightarrow{AB}$ represents the directed line from A to B. Direction vectors are represented by small letters to describe, for instance, the directed line from A to B as $\vec{a} = \overrightarrow{AB}$, while $\vec{a}^0$ describes the corresponding unit vector along $\vec{a}$. The angle φ of $\vec{a}$ is determined anticlockwise by the angle between $\vec{a}$ and $\vec{e_1}$.

4.2.2 *Area Partitioning*

For the correct operation BLOCKTREE.KOM depends on a logical partitioning of the considered area into *blocks*, relying on squares for the partitioning. As introduced in Section 4.1 the blocks are used to determine clusters of nodes, which serve for the establishment of the tree of blocks as the underlying monitoring topology. As defined in our previous work [157] the size of a block depends on BLOCKTREE.KOM's configurable system parameter r_{max}, which specifies the estimated maximum communication range of nodes. BLOCKTREE.KOM demands that a node reaches all other nodes in the same block within one hop. Consequently, the size of a block depends on the estimated maximum communication range and is calculated, as described in (5) and depicted in Figure 13a, where s_{block} represents the edge length of the square block. Depending on the estimated communication range r_{max}, it may happen that the effective communication range is smaller and that a node does not reach every other node in the same block. In this case it is not guaranteed that the monitored values of a node will be taken into consideration and reflected in the monitoring results. Contrary, a conservative estimation, which underestimates the effective com-



(a) Visualizing the dependency between the estimated communication range r_{max} and the resulting edge length s_{block} of the blocks

(b) Visualizing the difference between the coverage of a hexagon and a square, using the same diameter r_{max}

Figure 13: Visualization of different properties of the logical area partitioning based on the estimated communication range r_{max}

munication range, leads to smaller blocks and increases the unnecessary reception of information from foreign blocks.

$$s_{block} = \frac{r_{max}}{\sqrt{2}} \tag{5}$$

Contrary to other approaches [83], which propose to use hexagons as an alternative for splitting the area as a function of the communication range, BLOCKTREE.KOM relies on square blocks for the area partitioning. Though, hexagons exhibit an approximately 1.3 times larger coverage of the area for a given diameter in contrast to squares (cf. Figure 13b), the creation of a hierarchical topology based on hexagons results in complex polygons, which, for instance, complicate the computation of the affiliation to a certain polygon. For that reason many approaches, which rely on a hierarchy of areas and where a higher level consists of a federation of subjacent areas, simplify the establishment and use square blocks for the hierarchical topology. Examples for such approaches in MANETs range from hierarchical location services [101, 83] over replication techniques [99] to other decentralized monitoring mechanisms [51].

As defined in our previous work [157] a node K can determine its surrounding block, using its current position. Therefore, each block obtains a unique two-dimensional identifier ID_{block}, which is calculated based on (6). For the unique identification of blocks the nodes require a point of reference. Since the UTM system already splits the earth into zones, these zones, more precisely, the lower left points of zones serve as reference points to identify blocks. Given (6), the reference points, and s_{block}, a node is able to determine parent or child blocks in addition to the identification of its own block.

$$ID_{block} = \left(\left\lfloor \frac{K[1]}{s_{block}} \right\rfloor, \left\lfloor \frac{K[2]}{s_{block}} \right\rfloor \right) \tag{6}$$

4.3 COMMUNICATION METHODS

As mentioned in the beginning of this chapter BLOCKTREE.KOM relies on its own routing methods to enable the communication between the nodes. The decision results from the fact to avoid the dependency on a certain routing protocol and to operate if the underlying routing protocol fails, as stated by Nanda and Kotz [119]. Furthermore, the own routing methods are tailored to the characteristics of the service, i.e., monitoring of MANETs, and must not operate on service-agnostic routing protocols. The designed routing methods adopt ideas and concepts from geocast protocols, which belong to the class of position-based routing protocols and where the route from a source to a destination mainly depends on the positions[1] of nodes. In contrast to, e.g., topology-based routing protocols (cf. Section 2.3.2), they are considered to scale with the number of participating nodes [101, 66] and to be robust to the frequently changing topology of the MANET [40]. Moreover, we are rather interested in addressing and deliver data to geocast regions than to single nodes, which supports the selection of ideas from geocast protocols.

4.3.1 *Description of the General Routing Procedure*

BLOCKTREE.KOM introduces two classes of routing methods [157], which we denote as *receiver-based region geocast* and *receiver-based area dissemination* and which are detailed in Section 4.3.2 and 4.3.3, respectively. Both classes of routing methods comply with the communication concepts, as presented in Section 4.1.2, and involve multiple nodes for data transmission, comprising multiple sources, intermediate nodes, and receiving nodes. Furthermore, they rely on contention-based forwarding schemes, which have been introduced in Section 2.3.3.3. These schemes ensure a robust transmission through a highly dynamic network but avoid an increased message overhead and the congestion of the communication medium, for instance, due to broadcast storms [172]. In the following we (i) detail the integration of contention-based forwarding, (ii) introduce the concept of a *message forwarding cache* for the identification of redundant and unnecessary messages, and (iii) describe the implementation of initiating and forwarding a BLOCKTREE.KOM message.

Similar to existing contention-based forwarding schemes [40, 189, 53] a node selects neither one nor multiple next hops but forwards the message by broadcasting, which is inherent to MANETs. Some of the receiving nodes become candidates to forward the message if they currently reside in a certain area, e.g., the greedy area (cf. Figure 5). To determine the effective forwarding node each candidate calculates a *hesitation time* $t_{hesTime}$ based on (7) and as defined in our previous work [157].

$$t_{hesTime} = k_{hesFactor} * t_{maxForward}, \quad 0 \leq k_{hesFactor} \leq 1 \tag{7}$$

In this equation $t_{maxForward}$ is a configurable system parameter of BLOCKTREE.KOM, representing the constant *maximum forwarding time*, while the *hesitation factor* $k_{hesFactor}$ varies between zero and one and is calculated by each forwarding candidate. The determination of the set of forwarding candidates and the calculation of the hesitation

1 For the remainder of this chapter the position of a node always refers to its position in the Cartesian coordinate system after applying the coordinate transformation on its geographic coordinates (cf. Section 4.2.1).

time depend on the effective routing procedure and are introduced in Section 4.3.2 and 4.3.3, respectively. The candidate with the lowest hesitation time becomes the forwarding node. In contrast to other contention-based forwarding schemes [189] the forwarding node does not announce itself at the previous hop but immediately forwards the received message, using broadcasting as well. Other candidates, which overhear this transmission, drop their hesitating messages.

In addition to contention-based forwarding, which ensures that at least one node out of several candidates sends a message, BLOCKTREE.KOM introduces the message forwarding cache. This cache is maintained by every node and identifies redundant messages, which should not be processed and forwarded but deleted to save resources and avoid unnecessary relaying. Reasons for redundant messages may result from a delayed delivery, communication errors, or unawareness of recently sent messages. Consequently, these messages are mainly detected after broadcasting the original message. For the identification of such messages each message obtains an ID, represented by ID_{msg}, which depends, e.g., on the message type or the geographical origin of a message, but never on specific information related to a single node. If a message is received or created and the message forwarding cache does not contain the corresponding ID_{msg}, the message is processed and ID_{msg} is stored in the cache. Every subsequent message with the same ID_{msg} is dropped. After a certain amount of time, an ID_{msg} is purged from the message forwarding cache. To determine that moment every message contains a field *timeout*, which is stored together with the related ID_{msg}. During the description of C- and P-BLOCKTREE.KOM (cf. Section 4.4 and 4.5), we provide information about ID_{msg} with the corresponding timeout.

INITIATING A NEW BLOCKTREE.KOM MESSAGE

After detailing how BLOCKTREE.KOM messages are forwarded, using contention-based forwarding and the message forwarding cache, we describe the initiation of a message by one or a set of sources $\mathcal{S}$. Out of $\mathcal{S}$, a source S, which wants to create and send a message, first generates the corresponding ID_{msg}. Subsequently, S queries its message forwarding cache for that ID_{msg} to check if the corresponding message must be sent or if a neighbor already sent that message. If the cache does not contain ID_{msg}, the corresponding message with the required fields and data is created and sent, while the cache is updated. Table 6 lists the fields of a BLOCKTREE.KOM message that are required to properly handle and deliver the message at the destination. Based on this procedure, a set of sources $\mathcal{S}$ prepares the initiation of a message. At least one node effectively sends the message after its calculated hesitation time expired. If the node fails and cannot transmit the message other sources are available to replace that node.

HANDLING AN INCOMING BLOCKTREE.KOM MESSAGE

After the reception of an incoming message, a node checks if its message forwarding cache contains the received ID_{msg}. If not, the cache is updated with the received ID_{msg} and timeout and the message is further processed. To determine if a node should only forward the message or even process the transported monitoring data it must determine, if it belongs to the set of receivers $\mathcal{D}$. The determination of $\mathcal{D}$ as well as the required information about the destination depends on the utilized routing procedure and are detailed in Section 4.3.2 and 4.3.3, respectively. Independent of the outcome, the node also checks if it currently resides in a specified area to be-

Message field	Size	Description
Message type	1 Byte	Specifies the message type, which helps to determine how the data must be handled
Source position	16 Bytes	Contains the position of the source, which created the message
Last hop position	16 Bytes	Contains the position of the previous hop
Hop count	1 Byte	Contains the current hop count of the message
ID_{msg}	4 Bytes	Contains the unique message identifier to identify new or redundant messages. Its format depends on the concrete implementation of BLOCKTREE.KOM and is detailed in Section 4.4 and 4.5.
Timeout	8 Bytes	Specifies when to purge the related ID_{msg} from the message forwarding cache. Similar to ID_{msg}, concrete configurations are detailed in Section 4.4 and 4.5.
Destination	Varying	Specifies the destination of a message. Its format depends on the routing method, as specified in Section 4.3.2 and 4.3.3.

Table 6: Overview and description of the required fields of a generic BLOCKTREE.KOM message, which is completed by further fields that depend on the effective message type as well as the actual monitoring data.

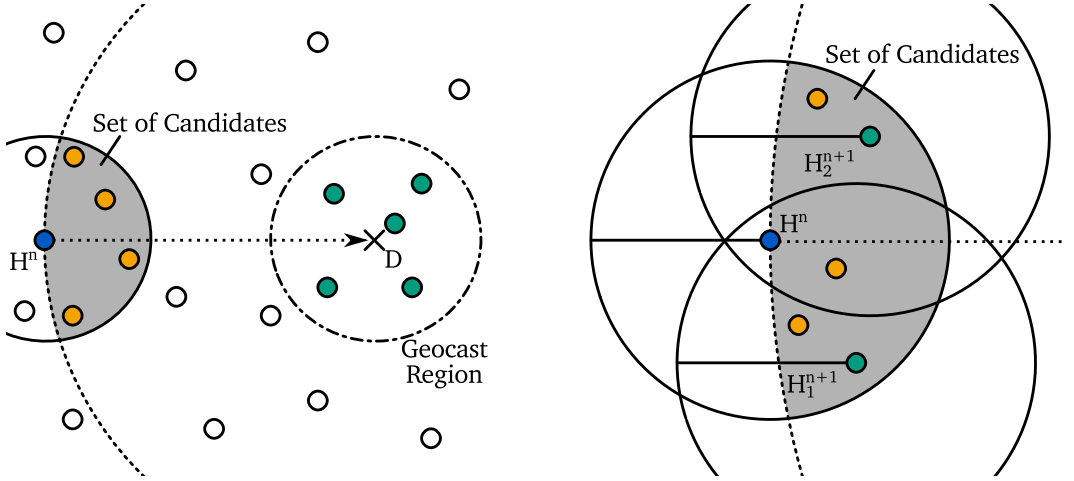
come a forwarding candidate for the message. If it belongs to the set of forwarding candidates, it subsequently determines its hesitation factor to calculate the resulting hesitation time based on (7). The hesitating message is dropped if the message is received from another node in the meantime, or it is sent after the hesitation time has elapsed.

4.3.2 *Receiver-Based Region Geocast*

After the description of the general routing procedure as well as initiating and forwarding a message, we present the concrete routing method that belongs to the class of receiver-based region geocast. This class of routing methods adopts concepts from data-transmission oriented geocast protocols, such as DREAM [7] or LAR [86]. It routes data through the network to a specified geocast region, where the set of intermediate nodes $\mathcal{I}$ constitutes the forwarding region. Particularly, we describe (i) the method to identify if a node currently resides in a specified area to become a forwarding candidate, (ii) the calculation of the hesitation factor, (iii) the definition of a destination, and (iv) the related method to determine if a node currently resides at the destination.

4.3.2.1 *Directed Target Region Geocast*

The *Directed Target Region Geocast* uses the *greedy area* to determine the set of forwarding candidates $\mathcal{C}$. The greedy area represents the fraction of a node's communication



(a) Visualization of the set of candidates $\mathcal{C}$ that comprises all neighbors of the broadcasting hop H^n

(b) Concurrent forwarding of a messages from two nodes, which cannot overhear each other

Figure 14: Visualization of different aspects of the Directed Target Region Geocast

range, which is closer to the destination, as specified in Section 2.3.1. Figure 14a depicts the resulting set $\mathcal{C}$, which comprises all neighbors of the broadcasting hop H^n, which reduce the distance to the destination. Given the set of all direct neighbors $\mathcal{N}$ of H^n, $\mathcal{C}$ is defined as

$$\mathcal{C} = \left\{ N \middle| N \in \mathcal{N} \wedge \left|\overrightarrow{ND}\right| \leq \left|\overrightarrow{H^n D}\right| \right\}. \tag{8}$$

Based on the information in a BlockTree.KOM message about the previous hop H^n and the destination D a receiving node checks if its distance to D is less or equal to the distance $\left|\overrightarrow{H^n D}\right|$. Dependent on the outcome it either belongs to $\mathcal{C}$ or not.

For the calculation of the hesitation factor $k_{hesFactor}$ and to determine if a node will forward the message and become an intermediate node I the Directed Target Region Geocast utilizes the Most Forward Within Radius scheme [166], as shown in (9).

$$k_{hesFactor} = \begin{cases} 0 & \text{if } \vec{c}\vec{d}^0 > r_{max} \\ 1 - \frac{\vec{c}\vec{d}^0}{r_{max}} & \text{else} \end{cases} \tag{9}$$

Given the normalized direction vector $\vec{d}^0$ of $\overrightarrow{H^n D}$ as well as $\vec{c}$, (9) calculates $k_{hesFactor}$ based on the projection of $\vec{c}$ on $\vec{d}^0$. The vector $\vec{c}$ represents the line from the broadcasting node H^n to the potential next hop H^{n+1} from $\mathcal{C}$. The projection is normalized to the interval $[0, 1]$ by dividing through r_{max} and finally subtracted from one so that a longer projection leads to a smaller hesitation factor and effective hesitation time. If the length of the projection exceeds r_{max}, $k_{hesFactor}$ is set to zero. A potential drawback of this strategy results from the determination of $\mathcal{C}$, because it might happen that multiple potential candidates, e.g., H_1^{n+1}, H_2^{n+1} as illustrated in Figure 14b, send a message in parallel as they cannot overhear each other. In contrast, the sketched spreading of a message may help to handle concave nodes and to successfully forward messages in sparsely populated regions. As introduced in Section 2.3.3.1 a concave node is a node that has no neighbor that reduces the distance to the destination.

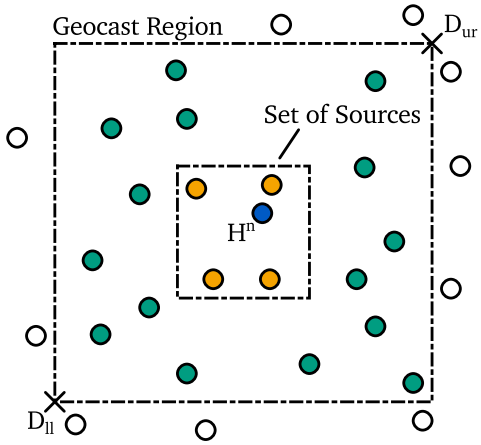


Figure 15: Example of a set of sources $\mathcal{S}$ inside a geocast region, which performs a Flooding-Based Area Dissemination.

For the definition of a destination the Directed Target Region Geocast uses a circular area to define the geocast region and the resulting set of receivers $\mathcal{D}$ of a message. To calculate this region the field destination of a BLOCKTREE.KOM message contains the position D that specifies the origin of the circular area. The corresponding radius depends on the estimated maximum communication range r_{max}, as depicted in Figure 14a.

Given the set of all active nodes $\mathcal{A}$, $\mathcal{D}$ is represented by (10) and helps to determine if a node K belongs to $\mathcal{D}$.

$$\mathcal{D} = \left\{ K \middle| K \in \mathcal{A} \wedge \left| \overrightarrow{KD} \right| \leq r_{max} \right\} \tag{10}$$

4.3.3 *Receiver-Based Area Dissemination*

Receiver-based area dissemination may be seen as a special case of receiver-based region geocast, since the corresponding routing method simply disseminates data inside a geocast region, which encompasses the current position of the set of sources $\mathcal{S}$ (cf. Figure 15). Analogously to Section 4.3.2, we detail (i) the identification of the set of candidates $\mathcal{C}$, (ii) the calculation of the hesitation factor, (iii) the definition of a destination, and (iv) the determination of a node's affiliation to $\mathcal{D}$.

4.3.3.1 *Flooding-Based Area Dissemination*

Contrary to the description of receiver-based region geocast, we start with a description of the message field destination, because it is required for the subsequent calculations. The information about the destination for the *Flooding-Based Area Dissemination* consists of two points, which are used to specify the geocast region that encompasses all receivers of a message. As depicted in Figure 15 the two points represent the lower left and the upper right corner of that region, which we denote as D_{ll} and D_{ur}, respectively.

For the determination of the set of forwarding candidates $\mathcal{C}$ the Flooding-Based Area Dissemination checks if the receiving neighbor of a broadcasted message by H^n currently resides in the geocast region (cf. Figure 15). Given the set of neighbors $\mathcal{N}$

of the broadcasting hop H^n, (11) is used to determine the set of candidates. The identification of forwarding candidates does not take into consideration if a node already sent a message or not, which would lead to continuous broadcasting of messages by every node. To prevent the extensive communication overhead BlockTree.KOM relies on the message forwarding cache, which detects redundant messages and drops them.

$$\begin{aligned}\mathcal{C} = \{N | N \in \mathcal{N} \wedge N[1] \geq D_{ll}[1] \wedge N[2] \geq D_{ll}[2] \\ \wedge N[1] \leq D_{ur}[1] \wedge N[2] \leq D_{ur}[2]\}\end{aligned} \tag{11}$$

The definition for the set of nodes $\mathcal{D}$, which currently sojourn at the destination area, is given in (12). The definition of $\mathcal{D}$ is similar to the definition of $\mathcal{C}$ in (11), except that it operates on the set of all active nodes $\mathcal{A}$ in the MANET instead of the set of neighbors.

$$\begin{aligned}\mathcal{D} = \{K | K \in \mathcal{A} \wedge K[1] \geq D_{ll}[1] \wedge K[2] \geq D_{ll}[2] \\ \wedge K[1] \leq D_{ur}[1] \wedge K[2] \leq D_{ur}[2]\}\end{aligned} \tag{12}$$

The corresponding calculation of the hesitation factor for the Flooding-Based Area Dissemination is defined in (13). Every receiving node H^{n+1} of the broadcasted message calculates its hesitation factor based on the distance to the transmitting node H^n. If the distance between H^{n+1} and H^n exceeds the estimated maximum communication range r_{max}, $k_{hesFactor}$ is set to zero. Otherwise, $k_{hesFactor}$ decreases for an increasing distance to H^n so that distant nodes are preferred to accelerate the dissemination of information using as less hops as possible to cover the whole geocast region.

$$k_{hesFactor} = \begin{cases} 0 & \text{if } \left|\overrightarrow{H^n H^{n+1}}\right| > r_{max} \\ 1 - \frac{\left|\overrightarrow{H^n H^{n+1}}\right|}{r_{max}} & \text{else} \end{cases} \tag{13}$$

4.3.4 *Summary*

After the description of the logical area partitioning and the different routing methods to communicate, Table 7 summarizes the common system parameters of BlockTree.KOM, which are required for the logical partitioning and the presented routing methods. Both approaches of BlockTree.KOM, presented below, must configure the listed system parameters to operate in MANETs. In addition, every approach of BlockTree.KOM defines its own set of additional system parameters, which are introduced during the presentation of the corresponding approach and summarized at the end of each section. The impact of BlockTree.KOM's general and approach-related system parameters on performance and cost will be evaluated in Section 6.3.2 as well as in Appendix A.2

System parameter	Symbol	Explanation
Maximum hierarchy level	l_{max}	Configures the maximum level of BlockTree.KOM's underlying hierarchy
Maximum forwarding time	$t_{maxForward}$	Specifies the constant factor to calculate the hesitation time to forward a message based on (7)
Estimated maximum communication range	r_{max}	Defines the estimated maximum communication range of nodes
Block size	s_{block}	Represents the edge length of a block, which depends on the estimated maximum communication range r_{max}
Position refresh interval	$t_{reqPosInterval}$	Determines after which amount of time a node retrieves and refreshes the information about its current geographical position

Table 7: Overview and description of general system parameters

4.4 C-BLOCKTREE.KOM: THE CONCENTRATING APPROACH

After the description of the logical area partitioning and communication methods, we introduce C-BlockTree.KOM. It represents one of the two approaches of BlockTree.KOM and relies on the fundamental topology and communication concepts, as introduced in Section 4.1. As presented in our previous work [157] and common for hierarchical monitoring mechanisms [100, 8] C-BlockTree.KOM relies on two separate phases for the collection of monitoring data and the dissemination of monitoring results. The hierarchical structure is exploited to collect and *concentrate* the data at given locations in the MANET, resulting in the concentrating approach with the related name of C-BlockTree.KOM. For this purpose the identified blocks of the network are arranged in a tree of blocks, which is discussed below. At every level l_x of the tree, there are several responsible blocks, which concentrate the data from lower levels and forward the aggregated data to concentrating blocks at the next level l_{x+1} of the tree. The data is collected along the tree of blocks until the highest level covers the whole MANET or the maximum hierarchy level l_{max} is reached. Contrary to the structured collection process of monitored data, the dissemination of the results is simply broadcasted by the concentrating block at the highest level of the hierarchy to the whole network. In the following we start with a description of the underlying hierarchy of C-BlockTree.KOM, taking the developed topology concepts into consideration. Subsequently, we describe how the tree of blocks is used for data collection and result dissemination, complying with the respective communication concepts.

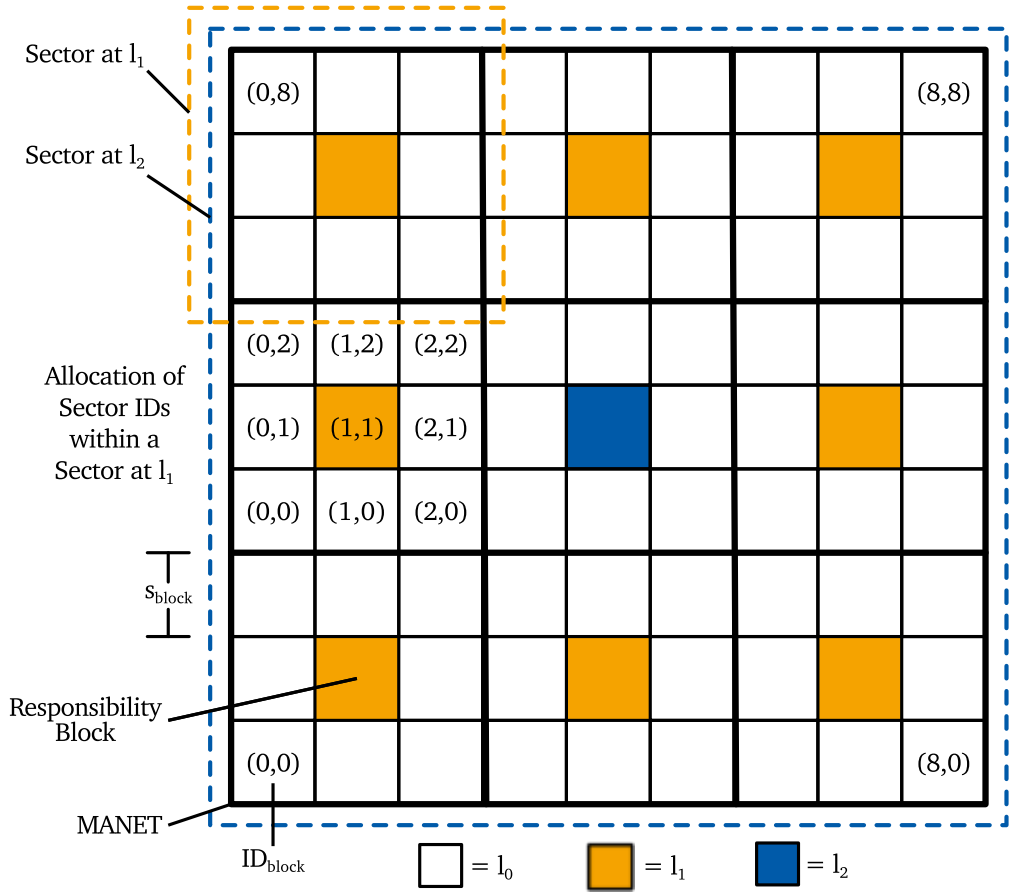


Figure 16: Creating the hierarchy of the concentrating approach with $l_{max} = 2$ (adapted from [157])

4.4.1 *Creating the Hierarchy*

To comply with the aforementioned topology concepts the underlying topology of C-BLOCKTREE.KOM ensures that parent-child relations always consist of multiple nodes. Therefore, C-BLOCKTREE.KOM establishes parent-child relations between blocks, leading to the tree of blocks. The concentrating approach uses the obtained blocks to define the locations for data concentration and processing. Nodes, which currently reside in one of these blocks, are in charge to collect, process, and forward the data to the next concentrating block at a higher level. With the selection of concentrating blocks C-BLOCKTREE.KOM tries to concentrate the data in the middle of the MANET so that the distance to a related, concentrating block does not differ among the nodes. Consequently, the highest block of the tree is ideally located at the center of the MANET and uniformly surrounded by blocks of lower levels, as depicted in Figure 16. To reduce the state information for the identification of relevant blocks C-BLOCKTREE.KOM just relies on the actual positions of nodes. In the following we elaborate on how nodes determine and calculate relevant blocks in the hierarchy, given their current position and their block.

Figure 16 depicts a fully established hierarchy of C-BLOCKTREE.KOM with $l_{max} = 2$ and highlights the concentrating blocks at the different levels, using colored blocks

in yellow (for l_1) and blue (for l_2). The concentrating blocks collect and process the data from a set of blocks and federation of blocks, which represent the corresponding *sector* of the concentrating block. In this context the concentrating block is also referred to as the *responsibility block* of that sector at level l_x. Figure 16 shows an example for a sector at l_1 as well as at l_2, which are surrounded by a dashed frame in yellow and blue. The width and height of the hierarchy mainly depend on the size of a sector, which determines the number of included blocks and lower sectors. Consequently, the edge length of a sector s_{sector} is measured in blocks or lower sectors and must be odd. Neglecting the current level of a sector and given the assumption that a sector is uniformly surrounded by blocks and lower sectors, s_{sector} is calculated as specified in (14). Figure 16 depicts an example for $n = 1$ and $s_{sector} = 3$.

$$s_{sector} = 2 * n + 1, \quad n > 0 \tag{14}$$

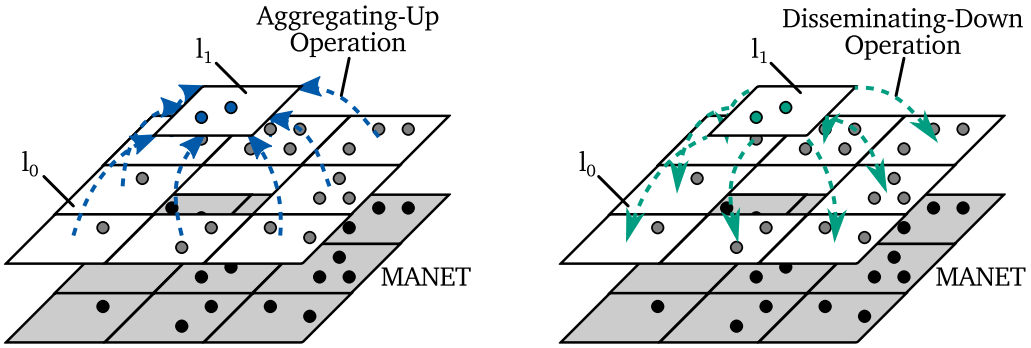
To create the hierarchy every block of the network belongs to one level of the hierarchy. Describing the creation of the hierarchy from the bottom, as detailed in our previous work [157], a block at l_0 consists of nodes. A concentrating or responsibility block at l_1 is surrounded by $s_{sector}^2 - 1$ blocks of l_0, forming a sector at l_1. A concentrating block, which is responsible for a sector at l_2, is also surrounded by $s_{sector}^2 - 1$ blocks at l_0 and additionally enclosed by $s_{sector}^2 - 1$ sectors at l_1 (cf. Figure 16). To generalize this property for concentrating blocks and sectors of higher levels l_x with $x \geq 2$ a concentrating block is always surrounded by $s_{sector}^2 - 1$ blocks at l_0 and $s_{sector}^2 - 1$ sectors at every l_y for $0 < y < x$. As already discussed and presented in our previous work [157] the expansion of the hierarchy either ends if the monitoring mechanism covers the whole MANET or if the specified maximum level l_{max} of the hierarchy is reached.

In addition to ID_{block}, which is used for the unique identification of blocks, every block or sector obtains another ID, which is used for the unique identification in the context of the next higher sector. The ID is referred to as ID_{sector} and described by

$$ID_{sector} = \left(\left\lfloor \frac{K[1]}{s_{block} * s_{sector}^{x-1}} \right\rfloor \bmod s_{sector}, \left\lfloor \frac{K[2]}{s_{block} * s_{sector}^{x-1}} \right\rfloor \bmod s_{sector} \right), \tag{15}$$

where $x \geq 1$ and $K[1]$ and $K[2]$ correspond to the position of node K, which currently resides at the respective block and sector. Figure 16 shows an example of the sector ID allocation for the sector in the middle on the left-hand side of the hierarchy.

After the description of the creation of the tree of blocks, which serves for data collection and result dissemination, we elaborate on how a node determines the level of its current block, just relying on its position. Given the position of a node K, K utilizes (6) to calculate ID_{block} of the block, where it currently resides. Based on ID_{block}, K determines the maximum level l_x of its block. Starting from $n = 0$, (16) is used to iteratively determine if a block belongs to level n, given the corresponding block ID. The equation returns the last argument n for which both conditions $\left(\left\lfloor \frac{s_{sector}^n}{2} \right\rfloor \equiv ID_{block}[1] \bmod s_{sector}^n \right)$ and $\left(\left\lfloor \frac{s_{sector}^n}{2} \right\rfloor \equiv ID_{block}[2] \bmod s_{sector}^n \right)$ hold. In addition to the determination of the level of the own block (16) is used to calculate the



(a) Execution of an AGGREGATING-UP operation from the surrounding blocks at l_0 to the concentrating block at l_1

(b) Execution of a DISSEMINATING-DOWN operation from a concentrating block at l_1

Figure 17: The underlying communication patterns of the concentrating approach

level of other blocks at any position. During the description of the data collection and result dissemination, we will draw on this functionality.

$$x = \arg\max_{n \in \mathbb{N}} \left(\left(\left\lfloor \frac{s^n_{\text{sector}}}{2} \right\rfloor \equiv \text{ID}_{\text{block}}[1] \bmod s^n_{\text{sector}} \right) \right.$$
$$\left. \wedge \left(\left\lfloor \frac{s^n_{\text{sector}}}{2} \right\rfloor \equiv \text{ID}_{\text{block}}[2] \bmod s^n_{\text{sector}} \right) \right) \tag{16}$$

4.4.2 *Data Collection and Result Dissemination*

After the description of the establishment of the underlying hierarchy and the determination of a block's level, the required operations for data collection and result dissemination are introduced. During the description, we explain how these operations implement the identified communication concepts, as specified in Section 4.1.2. In addition, we detail the utilization of the presented communication methods for the transmission of information, comprising, for instance, which methods are used and how the message forwarding cache is utilized.

As presented in our previous work [157], C-BLOCKTREE.KOM consists of the three operations LEAF-BROADCASTING, AGGREGATING-UP, and DISSEMINATING-DOWN. The first two operations serve for the data collection, whereas the last one is used to disseminate the results. Figure 17 sketches the corresponding communication patterns of the AGGREGATING-UP and DISSEMINATING-DOWN operations. The operations are periodically triggered by every node but do not require any type of synchronization between nodes so that C-BLOCKTREE.KOM operates completely unsynchronized. As a consequence, C-BLOCKTREE.KOM does not require specific operations to handle arriving or leaving nodes, instead, the nodes periodically execute the operations. The execution interval of all three operations is specified by the system parameter *aggregation interval* and represented by $t_{\text{aggInterval}}$.

Message field	Size	Description
Message type	1 byte	Specifies the message type, which helps to determine how the data must be handled
ID_{node}	8 bytes	Represents the current ID of the broadcasting node, which is never used for the direct addressing of a node but to distinguish between the received data from different nodes
ID_{block}	4 bytes	Stores the ID of a broadcasting node's block so that a receiving node is able to determine if the received LEAF-BROADCASTING message should be processed
Data	Varying	Contains the locally measured attributes per node. A monitored attribute is represented by an identifier with a size of one byte, while the data is represented as `double` with a size of eight bytes.

Table 8: Message format of a LEAF-BROADCASTING message

4.4.2.1 *Leaf-Broadcasting*

During the LEAF-BROADCASTING operation, a node broadcasts its locally measured monitoring attributes. The message is only intended for other nodes that currently reside in the same block as the broadcasting node. Since the size of a block depends on the estimated maximum communication range r_{max}, no additional routing method is required to reach all other nodes. The node broadcasts a LEAF-BROADCASTING message, which contains the locally measured attributes and the two additional fields ID_{node} and ID_{block}, which are explained in Table 8. The common fields of the generic BLOCKTREE.KOM message are not required in this context, because the LEAF-BROADCASTING message is just broadcasted over one hop and does not rely on any contention-based forwarding scheme. Consequently, the LEAF-BROADCASTING message contains its own field to provide information about the message type.

To decide if a broadcasted LEAF-BROADCASTING message should be processed a node calculates its ID_{block} based on its current position and checks if its ID_{block} matches the received ID_{block} of the message. If the IDs do not match, the message is dropped, otherwise the received data and ID_{node} are stored in the *leaf information table*. ID_{node} is only used to identify previously transmitted monitored attributes, which are overwritten if they exist. If no further information of that node has been received for a certain amount of time, the corresponding data are purged from the table. The timeout for the removal of data is specified by the system parameter $t_{leafInfoTimeout}$.

The leaf information table contains the monitored attributes from other nodes of the same block, which advertise their data through LEAF-BROADCASTING messages. Consequently, the current state of the block in terms of the monitored attributes is stored and summarized in this table, which serves as input for the AGGREGATING-UP operation, as detailed afterwards. Since every node maintains such a table, the AGGREGATING-UP operation may be triggered by every node so that the subsequent collection of data over the hierarchy does not depend on a single but on all present nodes in a block. If a node leaves its block and enters a new one, the node purges its table. In addition, the nodes from the previous block purge the corresponding

entry of that node from their tables. The associated procedures are presented in Section 4.4.3.

4.4.2.2 *Aggregating-Up*

The AGGREGATING-UP operation provides a parent block with the aggregated monitoring data of a child block. If a block at l_0 executes this operation, the aggregated monitoring data just consist of the data, which is currently present in the leaf information table of a node in that block. If a block at l_x with $x > 0$ executes this operation, the aggregated data comprise information from the leaf information table as well as from the so-called *hierarchy table* that holds data from the surrounding blocks and sectors. Figure 17a depicts an example of an AGGREGATING-UP operation, where the surrounding blocks send their data to the corresponding concentrating block. In the following we describe the identification of a parent block, detail the format and transmission of an AGGREGATING-UP message, describe the structure and utilization of the hierarchy table, and outline how nodes process incoming AGGREGATING-UP messages.

To determine the parent block for an AGGREGATING-UP message a node requires its current position and the level l_x of the block, where it currently resides. Based on this information, it determines the center C_{x+1} of its parent block at the higher level l_{x+1}. Regarding C_{x+1}, the subscript $x+1$ represents the level l_{x+1} of the concentrating block. In addition, the center C_{x+1} specifies the destination of the AGGREGATING-UP message. The center is calculated based on (17) and requires the position of a node K, its level l_x, and the edge length s_m of the surrounding sector at the next higher level l_{x+1}. The first addend in (17) specifies the lower left corner of the sector and the second addend identifies the center relative to the lower left corner. Contrary to the definition of s_{sector} in (14), which returns the length of a sector either in blocks or sectors, s_m calculates the length in meter using $s_m = s_{block} * s_{sector}^{x+1}$. In this equation x represents the corresponding level l_x of the block, where the node currently resides.

$$C_{x+1} = s_m * \left(\begin{bmatrix} \frac{K[1]}{s_m} \\ \frac{K[2]}{s_m} \end{bmatrix} \right) + \begin{pmatrix} \frac{s_m}{2} \\ \frac{s_m}{2} \end{pmatrix} \tag{17}$$

During every aggregation interval, a block (i) determines its parent block, (ii) executes an AGGREGATING-UP operation, and (iii) sends an AGGREGATING-UP message to the identified parent block. If the highest level of the hierarchy is below l_{max}, as C-BLOCKTREE.KOM already covers the whole MANET at a lower level, the responsible block at that level must still execute the operation and send a message to its parent block at the next higher level. Since the corresponding nodes in the block are not aware if there are nodes present at the targeted block, this message is either dropped because the destination cannot be reached or the message is successfully delivered at the destination incrementing the hierarchy by one level. The advantage of this approach is that the hierarchy autonomously increases or decreases and adapts to the spatial size of the network. However, it comes at the expense of an increased traffic overhead. If l_{max} contains a high value while the effective level of the hierarchy is lower, nodes constantly try to discover the next higher level without any success.

Inside a block, every node is in charge to execute the AGGREGATING-UP operation once per aggregation interval. To avoid that the execution at every node leads to a

Message field	Size	Description
Aggregating-Up message		
l_x	1 byte	Specifies the level of the source's block
ID_{sector}	1 byte	Identifies the sector of the source, corresponding either to a block or to a sector
Data	Varying	Contains the aggregated monitoring attributes, representing either the state of a block or a sector. The aggregate of an attribute consists of the identifier as well as fields to store the minimum, maximum, sum, and number of nodes, resulting in an overall size of 29 bytes per monitored attribute.
Generic BlockTree.KOM message		
ID_{msg}	4 bytes	The message ID corresponds to ID_{block} of the source S of the AGGREGATING-UP message and is stored in the message forwarding cache to avoid that multiple nodes from the considered block initiate redundant AGGREGATING-UP messages.
Timeout	8 bytes	Specifies how long a node keeps the corresponding ID_{msg} in its forwarding cache. For the AGGREGATING-UP message, the time interval is given by the system parameter AGGREGATING-UP blocking interval $t_{aggUpBlocking}$.

Table 9: Message format of an AGGREGATING-UP message in C-BLOCKTREE.KOM

massively congested communication medium the message forwarding cache is used. Based on the cache, a node determines if an operation should be executed and an AGGREGATING-UP message should be sent or if it should be aborted, since the required message has already been sent by another node. Exploiting this approach, we comply to our first communication concept (cf. Section 4.1.2), because every node in a block acts as a potential source of an AGGREGATING-UP message. This increases the probability that a message is sent to a parent block, while the number of transmitting nodes is reduced to ideally one node per block. Any further attempt to send an AGGREGATING-UP message during the same interval is blocked. For the transmission of the message to the circular geocast region receiver-based region geocast is used. Thus, the resulting AGGREGATING-UP message comprises the common fields of a generic BLOCKTREE.KOM message (cf. Section 4.3.1) in addition to the specific fields of an AGGREGATING-UP message. Table 9 lists and explains the specific fields of the AGGREGATING-UP message, also providing details for ID_{msg} and timeout of the generic BLOCKTREE.KOM message. As outlined both fields depend on the concrete implementation of BLOCKTREE.KOM and the considered operation.

As mentioned at the beginning of this section and depicted in Figure 17a a parent block receives the aggregated monitoring data of its child blocks. The aggregated data comprise the monitored attributes of ideally all nodes, which currently reside in the block. For C-BLOCKTREE.KOM the aggregate of an attribute consists of the

minimum, maximum, sum, number of nodes, and average. To comply with the hierarchical computation property the average is calculated based on the *sum* of an attribute and the *number of nodes* as $f_{avg} = {}^{sum}/_{(\#\,nodes)}$. As a result the corresponding fields of an aggregate just comprise minimum, maximum, sum, and number of nodes: the first three attributes are represented as `double`, whereas the number of nodes is represented as `integer`. Together with the identifier of one byte the size of an aggregated attribute comes to 29 bytes.

Given the example in Figure 16, receiving nodes, which belong to the concentrating block at l_2, store the monitoring data in the hierarchy table. This covers data (i) from the surrounding eight sectors, corresponding to single blocks at l_0, and (ii) from the eight sectors at l_1. The reason for storing data from several levels results from the fact that a block at l_x with $x \geq 2$ is always surrounded by sectors from multiple levels, as defined in Section 4.4.1. Consequently, a hierarchy table of a node at l_x stores the aggregated monitoring data from the surrounding sectors at every level l_y with $0 \leq y < x$. For $y = 0$ a sector corresponds to a block. Furthermore, nodes of the concentrating block still store the local values from other nodes of the same block in the leaf information table to represent the state of the block.

The two fields l_x and ID_{sector} of an AGGREGATING-UP message serve for the identification of the corresponding cell in the hierarchy table. Similar to the management of the leaf information table, older values in a cell are overwritten, while the values are purged after a certain amount of time. The timeout for the data is specified by the system parameter $t_{aggUpTimeout}$.

4.4.2.3 *Disseminating-Down*

The DISSEMINATING-DOWN operation disseminates the aggregated monitoring results from the concentrating blocks to the remaining blocks, which belong to the corresponding sector of the concentrating block. Figure 17b depicts an example for the execution of a DISSEMINATING-DOWN operation. Each node, which currently resides in a concentrating block at l_x with $x > 0$, may start this operation and disseminate its currently available results. To avoid that the network gets congested by redundant DISSEMINATING-DOWN messages from a block receiver-based area dissemination (cf. Section 4.3.3) is used as underlying routing method to spread the data in the specified area. During an aggregation interval, every node in a concentrating block prepares the dissemination, while only one node completes the operation and transmits the corresponding message. The addressed geocast area of the message corresponds to the sector of the concentrating and disseminating block. With a higher level of the disseminating block the covered geocast area increases, either covering the whole MANET at the highest level or only a fraction if the maximum hierarchy level l_{max} is reached. Position and size of the dissemination area are given by the two points D_{ll} and D_{ur}, which represent the lower left and upper right corner of the dissemination area. Given a node K and level l_x of the concentrating block, (18) and (19) specify D_{ll} and D_{ur} of the corresponding dissemination area. In these equations s_m calculates the length in meter using $s_m = s_{block} * s_{sector}^x$.

$$D_{ll} = s_m * \begin{pmatrix} \left\lfloor \frac{K[1]}{s_m} \right\rfloor \\ \left\lfloor \frac{K[2]}{s_m} \right\rfloor \end{pmatrix} \tag{18}$$

$$D_{ur} = s_m * \begin{pmatrix} \left\lfloor \frac{K[1]}{s_m} \right\rfloor \\ \left\lfloor \frac{K[2]}{s_m} \right\rfloor \end{pmatrix} + \begin{pmatrix} s_m \\ s_m \end{pmatrix} \tag{19}$$

For the provisioning of location-aware monitoring results a concentrating block does not only disseminate the monitoring results to the corresponding sector at the highest level of the block. In addition it disseminates the monitoring results to sectors at lower levels for which it is responsible as well. Given the example of C-BlockTree.KOM in Figure 16, the concentrating block at l_2 disseminates the results (i) from l_1 to its eight surrounding blocks and (ii) from l_2 to the whole MANET. Consequently, a Disseminating-Down message specifies multiple dissemination areas in addition to the basic destination, which is required by the generic BlockTree.KOM message.

Table 10 lists the required fields of the resulting Disseminating-Down message, containing (i) the specific fields of that message as well as (ii) the common fields of the generic BlockTree.KOM message.

- The two arrays l[] and destinations[] specify the dissemination areas with the corresponding levels.
- The transmitted data comprise the monitoring results for the specified dissemination areas.
- The fields of the generic BlockTree.KOM message are required to execute the receiver-based area dissemination (cf. Section 4.3.1) and define how long ID_{block} is stored in the message forwarding cache to identify redundant and old Disseminating-Down messages.

A receiving node of a Disseminating-Down message stores the obtained results in its *result table*. The received information about the covered levels of the concentrating block is used to store the monitoring results from these levels in the result table, which stores data from l_1 up to l_x with $x \leq l_{max}$. Existing results in the table are updated, whereas older values are purged after a certain amount of time, which is specified by the system parameter $t_{dissDownTimeout}$. Based on this configuration, the result table of C-BlockTree.KOM provides location-aware monitoring results, because the lowest level of the table contains detailed results of a node's vicinity and higher levels provide a summarized view over larger areas. To obtain information about its current block a node consults its leaf information table.

4.4.3 *Handling the Dynamics of MANETs*

After the description of the monitoring topology and the operations to collect and disseminate monitoring information, we discuss how C-BlockTree.KOM handles the dynamic nature of MANETs, comprising node mobility and churn. Due to BlockTree.KOM's basic concepts C-BlockTree.KOM is already designed to handle churn, because neither the topology nor the communication relies on parent-child relations between two single nodes. Moreover, the unsynchronized periodic execution of the information exchange and the nearly stateless design do not require distinct methods

Message field	Size	Description
Disseminating-Down message		
$l[]$	$l_x * 1$ byte	Contains the levels of the related sectors from the concentrating block in an array. For these levels, the related results are disseminated. The highest level of the corresponding block is given by l_x also representing the size of the corresponding array.
Destinations[]	$(l_x - 1) *$ 32 bytes	Contains the specifications of the dissemination areas, where the related results are disseminated. The specification of the dissemination area for the highest level l_x is already specified in the generic BLOCKTREE.KOM message and not required in this context.
Data	Varying	Contains the monitoring results that represent the current state of the covered sectors from the concentrating block. The monitoring results consist of the aggregated attributes.
Generic BlockTree.KOM message		
ID_{msg}	4 bytes	The message ID corresponds to the ID_{block} of the source S of the DISSEMINATING-DOWN message. The ID is stored in the message forwarding cache to avoid that multiple nodes from the disseminating block initiate redundant DISSEMINATING-DOWN messages.
Timeout	8 bytes	Specifies how long a node keeps the corresponding ID_{msg} in its forwarding cache. For the DISSEMINATING-DOWN message, the time interval is given by the system parameter DISSEMINATING-DOWN blocking interval $t_{dissDownBlocking}$.

Table 10: Message format of a DISSEMINATING-DOWN message

to handle arriving or leaving nodes. Despite the good ability to handle churn, extensions are required to handle the mobility of nodes. Particularly, we must ensure that the concentrated data remain in the concentrating blocks and do not leave the blocks with the moving nodes. Otherwise, an executed AGGREGATING-UP or DISSEMINATING-DOWN operation relies on incomplete information for data collection or result dissemination. Furthermore, a moving node, which enters a new block, must not use the data from the old block or even from a sector to represent the state of the current block or sector. Below, we detail the additionally required functionality to tackle both problems.

To deal with the first problem every block at l_x with $x > 0$ utilizes the LEAF-BROADCASTING operation to replicate the data from the hierarchy table. Thus, nodes do not only broadcast their locally measured monitoring attributes but also the partial

aggregation results from the surrounding blocks and sectors. A replication message relies on the same message fields as the LEAF-BROADCASTING message, except that the data differ. The additional replication of data leads to an increased traffic. However, the replication prevents the loss of monitoring information from the surrounding blocks and sectors if a node leaves the concentrating block. Instead, the monitoring data reside at the nodes inside the concentrating blocks, which can operate on the entire data when starting an AGGREGATING-UP or DISSEMINATING-DOWN operation.

To tackle the second problem and to avoid that monitoring data from a previous block are used to represent the state of the current block a node deletes its leaf information table and its hierarchy table if it notices that it entered another block. After the update of its own state, it immediately sends a LEAF-BROADCASTING message irrespective of the aggregation interval. The message is used to update other nodes of the new block, whereas nodes from the previous block purge the corresponding entry from their leaf information table if they receive that message. Moreover, a node is not allowed to execute an AGGREGATING-UP or DISSEMINATING-DOWN operation for a certain time interval, which is specified by the system parameter *operation blocking interval* $t_{blockOperation}$. Based on this procedure the probability is increased that operations are executed by nodes that are equipped with sufficient information.

4.4.4 *Summary*

C-BLOCKTREE.KOM is a hierarchical decentralized monitoring mechanism for MANETs, which operates on a tree of blocks to collect monitoring data and disseminate the results. To manage and transmit the monitoring information C-BLOCKTREE.KOM consists of three operations, which are periodically executed.

- LEAF-BROADCASTING disseminates the locally measured data of a node inside its block.
- AGGREGATING-UP pushes the data of the populated blocks towards the center of the MANET. The operation relies on specific blocks, which are in charge of concentrating and processing the data.
- DISSEMINATING-DOWN is executed by the concentrating blocks to disseminate the generated results in the corresponding sectors, which belong to the concentrating block at a given level.

Based on BLOCKTREE.KOM's basic concepts as well as on the dedicated extensions, which address the characteristics of MANETs and the wireless communication medium, C-BLOCKTREE.KOM is designed to operate in dynamic environments providing accurate and timely results. Table 11 lists the specific, supplementary system parameters of C-BLOCKTREE.KOM, which are used to configure C-BLOCKTREE.KOM for its deployment in MANETs.

4.5 P-BLOCKTREE.KOM: THE PLANAR APPROACH

P-BLOCKTREE.KOM represents the planar approach of BLOCKTREE.KOM and simplifies the establishment of the topology as well as of the operations to communicate.

System parameter	Symbol	Explanation
Sector edge length	s_{sector}	Specifies how many sectors are federated along each edge
Aggregation interval	$t_{aggInterval}$	Defines the time interval between two consecutive executions of LEAF-BROADCASTING, AGGREGATING-UP, or DISSEMINATING-DOWN
Leaf information timeout	$t_{leafInfoTimeout}$	Determines after which amount of time the data, which belong to a certain node, are removed from the leaf information table
AGGREGATING-UP timeout	$t_{aggUpTimeout}$	Configures after which amount of time the data, which belong to a specific block or sector, are purged from the hierarchy table
AGGREGATING-UP blocking interval	$t_{aggUpBlocking}$	Specifies how long the message ID from an AGGREGATING-UP message is stored in the message forwarding cache to block redundant or old AGGREGATING-UP messages
DISSEMINATING-DOWN timeout	$t_{dissDownTimeout}$	Configures after which amount of time the monitoring results of a sector at the corresponding level are purged from the result table
DISSEMINATING-DOWN blocking interval	$t_{dissDownBlocking}$	Specifies how long the message ID from a DISSEMINATING-DOWN message is stored in the message forwarding cache to block redundant or old DISSEMINATING-DOWN messages
Operation blocking interval	$t_{blockOperation}$	Defines how long a node may not actively trigger AGGREGATING-UP and DISSEMINATING-DOWN operations after it entered a new block

Table 11: Overview and description of C-BLOCKTREE.KOM's specific system parameters

In contrast to C-BLOCKTREE.KOM the planar approach, as presented in our previous work [157], even increases the redundancy of data and incorporates more nodes in the communication process to handle the arising challenges of MANETs while providing accurate monitoring results. Consequently, P-BLOCKTREE.KOM relies on federations of blocks instead of single blocks to trigger the information exchange for two reasons.

- Reduction of the negative influence of sparsely populated areas to avoid sparsely populated blocks and prevent data loss

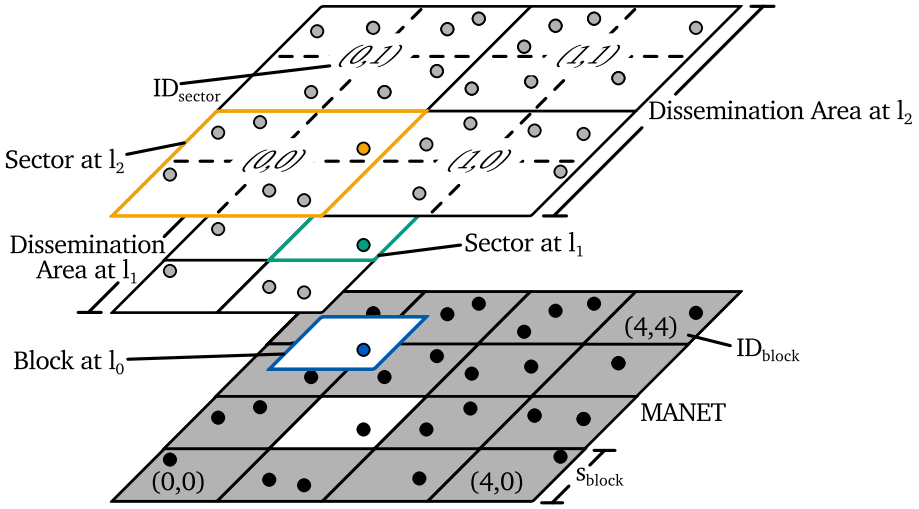


Figure 18: Creation of the hierarchy of the planar approach with $l_{max} = 2$ (adapted from [157])

- Reduction of the probability that a block loses the concentrated data due to moving nodes, which carry the data out of the block

Starting with a brief overview on the topology, the monitoring topology of P-BLOCKTREE.KOM may also be considered as a tree of blocks but the identification and, particularly, the arrangement of blocks differ as compared to C-BLOCKTREE.KOM. A block does not belong to a single but to every level of the hierarchy. This results from the fact that blocks from a lower level are federated at higher levels of the tree. The resulting hierarchy is depicted in Figure 18 and comparable to the hierarchy of Grid Box Hierarchy [51] or Astrolabe [176]. Dealing with the communication and operations for the information exchange, P-BLOCKTREE.KOM exclusively uses receiver-based area dissemination to increase data redundancy and include more nodes into the communication process. Moreover, the planar approach only consists of one phase and does not separately collect and disseminate information. During the collection phase, the data are simultaneously distributed so that no separate dissemination of the monitoring results is required. In the following we detail the monitoring topology and describe the collection of monitoring data and the dissemination of monitoring results.

4.5.1 *Creating the Hierarchy*

According to the presented topology concepts, the topology of P-BLOCKTREE.KOM relies on relations between single blocks or between federations of blocks. While C-BLOCKTREE.KOM uses these relations to define parent and child blocks at different levels, P-BLOCKTREE.KOM defines relations between sibling blocks or between sibling federations of blocks at the same level. Together, these sibling blocks or federations of blocks are federated to one larger block at the next higher level. Based on the rather planar relations between blocks, the corresponding nodes do not concentrate data at specified locations but disseminate data between sibling blocks. Similar to C-BLOCKTREE.KOM, a participating node of the planar approach just uses its actual position to determine its own block and other relevant blocks for the information

exchange. On the basis of the following example, we (i) introduce the nomenclature of P-BlockTree.KOM's hierarchy, (ii) present the establishment of the hierarchy, and (iii) subsequently detail the determination of relevant blocks.

Figure 18 presents an example of an established hierarchy of P-BlockTree.KOM with $l_{max} = 2$. The figure shows that the hierarchy consists of blocks at the lowest level l_0, which are obtained by the logical area partitioning. At l_1, a block from l_0 is denoted as a *sector*. Together with a set of sibling sectors at l_1 they define a *dissemination area* at the same level. The resulting shape and size of the dissemination area is a square and, similar to C-BlockTree.KOM, its size is given by the system parameter s_{sector}, which specifies how many sibling sectors are federated along each edge of a dissemination area. In addition, the sibling sectors of the dissemination area are used to determine a sector at the next higher level. Consequently, the dissemination area of a lower level is congruent to a single sector at the next higher level. As displayed in Figure 18 the dissemination area at l_1 corresponds to a sector at l_2. This procedure is repeated until the whole MANET is covered or l_{max} is reached. To differentiate between sibling sectors in the context of a sector at the next higher level P-BlockTree.KOM also relies on the two-dimensional identifier ID_{sector}. Given a node K, the system parameters s_{sector} and s_{block}, and the considered level l_x with $x \geq 1$, K relies on (15) to calculate its ID_{sector}.

Due to the fact that a node does not need to determine (i) the highest level of its block and (ii) the location of a concentrating block, the necessary computations of the monitoring topology are limited to two computations in addition to the calculation of ID_{sector}. In fact, a node just calculates the position and size of the dissemination area at a given level l_x. The obtained information serves for the Aggregating-Up operation and defines the boundaries for the dissemination of monitoring data (cf. Section 4.5.2.1). Position and size of the dissemination area are given by the two points D_{ll} and D_{ur}. As for C-BlockTree.KOM they represent the lower left and upper right corner of the dissemination area. Given the node K, level l_x of the corresponding dissemination area, and $s_m = s_{block} * s_{sector}^x$, (18) and (19) are used to determine D_{ll} and D_{ur}, respectively.

4.5.2 *Data Collection and Result Dissemination*

For the data collection and result dissemination P-BlockTree.KOM borrows the operations and concepts from C-BlockTree.KOM where possible. Similar to C-BlockTree.KOM P-BlockTree.KOM operates completely unsynchronized. Every node periodically executes its operations without the need to synchronize with other nodes so that no specific operations are required to handle arriving or leaving nodes. As for C-BlockTree.KOM the system parameter aggregation interval $t_{aggInterval}$ defines the interval for the periodic execution of the operations.

Dealing with the information exchange, P-BlockTree.KOM relies only on two instead of three operations to implement the communication. The Aggregating-Up operation already disseminates the monitoring results, which renders a dedicated operation for the result dissemination unnecessary. The Leaf-Broadcasting operation is borrowed from C-BlockTree.KOM, because the planar approach demands as well that all nodes from the same block are reachable within one hop and are aware of the currently monitored values of the neighbors. Consequently, each node main-

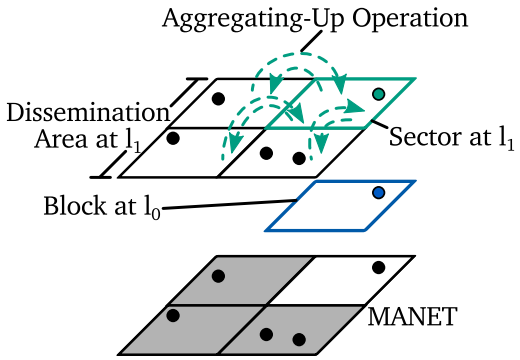


Figure 19: Execution of an AGGREGATING-UP operation of the green sector at l_1. The example depicts the message exchange of the green sector with sibling sectors in the resulting dissemination area at l_1.

tains a leaf information table, which stores the monitored attributes per node. For this table the leaf information timeout $t_{leafInfoTimeout}$ specifies when to purge old data. In contrast the AGGREGATING-UP operation is revised, because the communication takes place over a different hierarchy and results are simultaneously disseminated, while data are collected. In the following section we detail the AGGREGATING-UP operation for P-BLOCKTREE.KOM and complete the presentation of the planar approach with a description about handling the dynamic nature of MANETs.

4.5.2.1 *Aggregating-Up*

In general the AGGREGATING-UP operation is used to send the aggregated monitoring data to a specified area, where the data are collected and further processed. While C-BLOCKTREE.KOM relies on parent-child relations, where the parent corresponds to a single block that concentrates the data, the AGGREGATING-UP operation of P-BLOCKTREE.KOM disseminates data in the calculated dissemination area per level (cf. Figure 19). Due to the fact that a node, more precisely, the block of a node does not belong to a single but all active levels the operation is triggered for every active level. For the identification of active levels, a node keeps track of the received AGGREGATING-UP messages and uses the corresponding levels to determine the maximum level. All levels up to the observed maximum level are considered as active. Consequently, a node receives AGGREGATING-UP messages at a certain level if at least another sector from the same dissemination area at that level is populated. For the identification of a new level, for instance, if the spatial size of the MANET increases and covers a larger area a node periodically starts an AGGREGATING-UP operation at level l_{x+1} if l_x currently represents the highest active level. Relying on the example in Figure 18, which depicts an established hierarchy of P-BLOCKTREE.KOM with $l_{max} = 2$, a node starts the operation two times at l_1 and l_2. If $l_{max} \geq 3$, a node additionally starts the operation at l_3 to identify new sibling sectors at that level. To avoid that a node starts multiple AGGREGATING-UP operations simultaneously they are evenly distributed over the aggregation interval, starting from the lowest to the highest level.

During an AGGREGATING-UP operation, a node sends its aggregated monitoring data to the remaining sectors of a dissemination area so that the other sectors collect

Message field	Size	Description
Aggregating-Up message		
l_x	1 byte	Specifies the level of the source's sector
ID_{sector}	1 byte	Identifies the sector of the source
Data	Varying	Contains the aggregated monitoring attributes representing the state of a sector. The aggregate of an attribute consists of the identifier as well as fields to store the minimum, maximum, sum, and number of nodes, resulting in an overall size of 29 bytes per monitored attribute.
Generic BlockTree.KOM message		
ID_{msg}	4 bytes	The message ID consists of a hash of D_{ll} and D_{ur}, which is subsequently combined with ID_{sector} of the source S of the AGGREGATING-UP message. ID_{msg} is stored in the message forwarding cache to avoid that multiple nodes from the considered sector initiate redundant AGGREGATING-UP messages.
Timeout	8 bytes	Specifies how long a node keeps the corresponding ID_{msg} in its forwarding cache. Similar to C-BLOCKTREE.KOM the time interval to block AGGREGATING-UP messages is given by the system parameter AGGREGATING-UP blocking interval $t_{aggUpBlocking}$.

Table 12: Message format of an AGGREGATING-UP message in P-BLOCKTREE.KOM

and process the data. In turn and as depicted in Figure 19, the node receives the aggregated monitoring data from other sectors. On the basis of its own and the received data, a node can already calculate the monitoring results and is aware of the actual state of its sibling sectors in the dissemination area.

To identify the dissemination area for the own aggregated monitoring data (18) and (19) are used, which determine the lower left and upper right corner of the dissemination area. For the concrete transmission of the data receiver-based area dissemination (cf. Section 4.3.3) is used as underlying routing method. Similar to the AGGREGATING-UP operation in C-BLOCKTREE.KOM, every node of a sector prepares the transmission of an AGGREGATING-UP message per level, while only a fraction of nodes finally transmits the message.

Table 12 lists the required fields of the corresponding AGGREGATING-UP message, comprising the operation-specific fields, the monitoring data, and the common fields of a generic BLOCKTREE.KOM message. The common fields of the generic BLOCKTREE.KOM message comprise ID_{msg} and timeout and serve for the unique identification of AGGREGATING-UP messages to prevent the forwarding of redundant or old messages, due to the utilization of the receiver-based area dissemination. ID_{msg} is calculated based on (i) the two points D_{ll} and D_{ur} of the corresponding dissemination area and on (ii) ID_{sector} of the sending sector. To limit the size of the correspond-

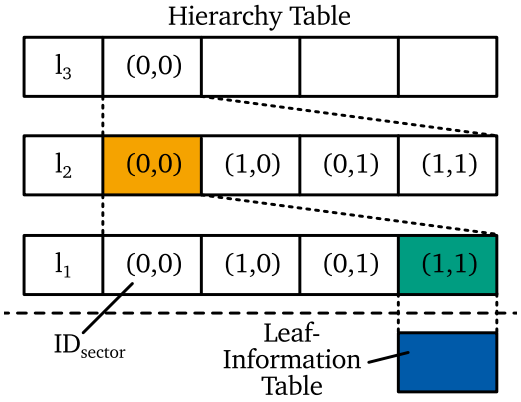


Figure 20: Example for a hierarchy table that represents the current state of the colored node in Figure 18 (adapted from [157])

ing ID_{msg} to four bytes a hash of the two points is generated and combined with ID_{sector}, using the XOR-operator. The resulting ID_{msg} is stored in the message forwarding cache for a certain amount of time, which is specified by the system parameter AGGREGATING-UP blocking interval $t_{aggUpBlocking}$. Based on this identifier, only a fraction of nodes of a sector finalizes the AGGREGATING-UP operation and sends a message, whereas the remaining nodes abort the operation. Furthermore, redundant messages of a sector are dropped by the receiving nodes.

The data of an AGGREGATING-UP message consist of the aggregated monitored attributes, which have been measured by the nodes that currently reside in the considered sector. For the specification of an attribute's aggregate we rely on the definitions as stated for C-BLOCKTREE.KOM. The aggregate of an attribute consists of the minimum, maximum, sum, number of nodes, and average, complying with the hierarchical computation property. Consequently, the corresponding fields of an aggregate just comprise minimum, maximum, sum, and number of nodes.

The operation-specific fields of an AGGREGATING-UP message comprise l_x and ID_{sector} of the message's source. Both fields are required to identify the appropriate cell in the hierarchy table, which stores the transmitted data. Existing values in a cell are overwritten with new information, whereas old values are purged after a certain amount of time. Similar to C-BLOCKTREE.KOM, the timeout for the data is given by $t_{aggUpTimeout}$. As depicted in Figure 20 and mentioned in our previous work [157] the table contains a row for every active level, which is split into cells according to the number of sectors per level. Monitoring data from sibling sectors are stored in the same row, while data of the own sector are recursively generated on demand. The data of a row serve as input for the sector at the next higher level. Figure 20 depicts an example of the hierarchy table, which belongs to the blue node that is shown in Figure 18. The shaded cells represent the sectors at l_1 and l_2, where the node currently resides. The result of a shaded cell at l_x consists of the four aggregated results from l_{x-1}.

4.5.3 *Handling the Dynamics of MANETs*

Similar to the description of C-BLOCKTREE.KOM this section details the extensions of P-BLOCKTREE.KOM with a focus on node mobility, because P-BLOCKTREE.KOM

is already designed to handle arriving and leaving nodes. As P-BlockTree.KOM borrows the Leaf-Broadcasting operation from C-BlockTree.KOM it also adapts the extensions to cope with moving nodes that enter a new block and leave the previous one. If a new block is detected, nodes immediately transmit a Leaf-Broadcasting message, while the receiving nodes in the former block use the received information to delete the corresponding entry of that node in their leaf information table. Moreover, leaving nodes do not execute an Aggregating-Up operation for a certain amount of time, which is specified by the system parameter *operation blocking interval* $t_{blockOperation}$. The blocking of operations increases the probability that an Aggregating-Up operation is only executed by nodes, which dispose of sufficient data to represent the current state of a sector. Besides the presented extensions that affect the communication, a node completely purges its hierarchy table to avoid that data from old sectors are mixed up with data from new sectors and falsify the state of the sectors. To prevent that a node just relies on its local information to assess the state of the MANET the *result caching table* is introduced, which serves as an alternative until the hierarchy table is filled with data from neighboring sectors. The result caching table may be compared to C-BlockTree.KOM's result table, because it consists of multiple entries that hold aggregated monitoring data of the different hierarchy levels. After the detection of a new block, a node aggregates the data per row of the hierarchy table and stores the obtained results in the result caching table in the corresponding row. For a certain amount of time, which is also specified by $t_{blockOperation}$, a node does not rely on its hierarchy table but on the result caching table to generate the state of the MANET. Similar to the result table of C-BlockTree.KOM the result caching table overwrites existing entries with recent results and purges stale entries after a certain amount of time. The duration until the deletion of stale entries is specified by the *result caching timeout* $t_{resCacheTimeout}$.

In contrast to C-BlockTree.KOM P-BlockTree.KOM abstains from replicating collected monitoring data at sectors of a higher level. As aggregated monitoring data at higher levels are not concentrated in single blocks but distributed over federations of sectors, there is no need for replication to keep the monitoring data in these areas. Due to the increasing size of the federations of sectors the probability decreases that every node with the associated monitoring data leaves this area.

4.5.4 *Summary*

P-BlockTree.KOM constitutes a hierarchical decentralized monitoring mechanism for MANETs that operates on a tree of blocks for the collection of monitoring data and dissemination of monitoring results. Contrary to C-BlockTree.KOM, the blocks of the tree do not belong to a single but to any active level of the hierarchy, because the blocks are federated to sectors at higher levels. To manage and transmit the data P-BlockTree.KOM only requires two instead of three operations.

- Leaf-Broadcasting is borrowed from C-BlockTree.KOM to disseminate the locally measured data inside a block.
- Aggregating-Up exchanges information with sibling sectors of the same level. The information exchange between sibling sectors has the effect that the collection of monitoring data leads to the immediate provisioning of monitoring results at the nodes.

System parameter	Symbol	Explanation
Sector edge length	s_{sector}	Specifies how many sectors are federated along each edge (i) to form a dissemination area at the same level l_x or (ii) to define a new sector at the next higher level l_{x+1} with $x \geq 1$
Aggregation interval	$t_{aggInterval}$	Defines the time interval between two consecutive executions of LEAF-BROADCASTING or AGGREGATING-UP
Leaf information timeout	$t_{leafInfoTimeout}$	Determines after which amount of time the data, which belong to a certain node, are removed from the leaf information table
AGGREGATING-UP timeout	$t_{aggUpTimeout}$	Configures after which amount of time the data, which belong to a specific sector, are purged from the hierarchy table
AGGREGATING-UP blocking interval	$t_{aggUpBlocking}$	Specifies how long the message ID from an AGGREGATING-UP message is stored in the message forwarding cache to block redundant or old AGGREGATING-UP messages
Operation blocking interval	$t_{blockOperation}$	Defines how long a node may not actively trigger AGGREGATING-UP operations after it entered a new block
Result caching timeout	$t_{resCacheTimeout}$	Configures after which amount of time the aggregated monitoring data, which represent the state of a sector at a certain level, are purged from the result caching table

Table 13: Overview and description of P-BLOCKTREE.KOM's specific system parameters

Considering the extensions of P-BLOCKTREE.KOM that address the mobility of nodes the approach adapts the extensions of C-BLOCKTREE.KOM but abstains from replicating collected monitoring data at higher levels. Moreover, the result caching table is introduced to prevent the assessment of the state of the network based on locally measured attributes. Table 13 lists the specific system parameters of P-BLOCKTREE.KOM in addition to the common system parameters of BLOCKTREE.KOM (cf. Table 7 in Section 4.3.4).

5 MOBI-G.KOM: FLAT AND GOSSIP-BASED MONITORING

After the description of the hierarchical decentralized monitoring mechanism BlockTree.KOM, this chapter introduces our second solution, referred to as Mobi-G.KOM. Similar to BlockTree.KOM Mobi-G.KOM is designed for the deployment in MANETs to provide the participating nodes with monitoring information, which serves as basis for the execution of transitions or multi-mechanisms adaptations to adapt the network, as envisaged by MAKI [49].

Mobi-G.KOM integrates all participating nodes into the monitoring process, where the nodes actively monitor the specified attributes and exchange their information. In contrast to the previously presented hierarchical approach Mobi-G.KOM operates on a flat and arbitrary topology using gossiping to exchange information between the nodes. The gossip-based communication pattern led to the name Mobi-G.KOM, which is an abbreviation for *mobile gossiping* and characterizes how the mobile nodes exchange information to monitor the MANET.

Without an underlying hierarchical topology we sacrifice the possibility to provide location-aware monitoring results for two main reasons: First, Mobi-G.KOM must not manage a logical hierarchy to collect data and disseminate information in MANETs. Based on this decision, we investigate the effect on performance and cost of a decentralized monitoring mechanism for MANETs if (i) no hierarchy must be established and managed and (ii) the specifications for a message's source and, particularly, destination are relaxed. Second, the developed approach for MANETs applies gossiping as a robust communication pattern to exchange the information between nodes. To avoid the problem of mass loss, which becomes particularly apparent if exchanged information between nodes gets lost, Mobi-G.KOM seizes the idea of population algorithms [5] for the collection and processing of data. Similar to Gossipico [174] or the Two-Phases algorithm [50] Mobi-G.KOM uses gossiping to communicate and abstains from gossiping to aggregate, as performed by other well-known gossip-based approaches [69, 170, 181]. Thus, Mobi-G.KOM relies on the

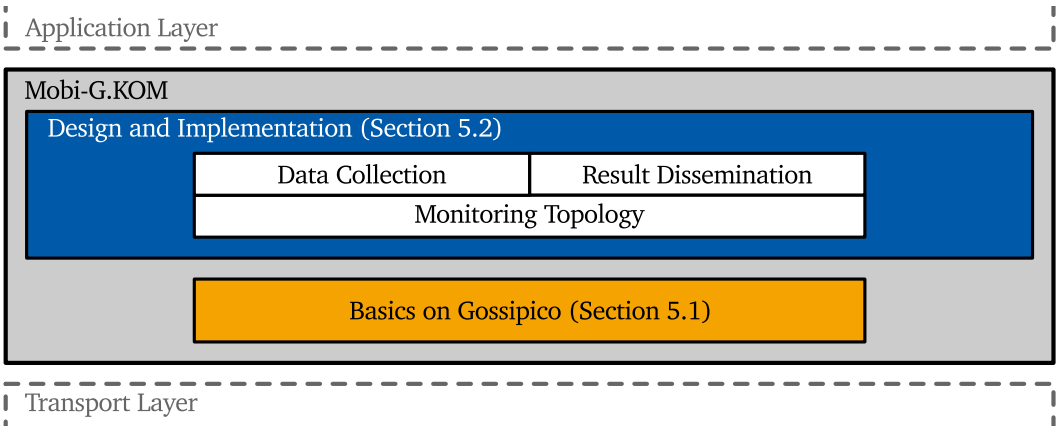


Figure 21: Overview on Mobi-G.KOM

hierarchical computation property instead of mass conservation to process and aggregate the monitored information, as detailed in the following sections. As already stressed Mobi-G.KOM does not operate on top of a hierarchy, which is configurable in its height and specifies the monitoring scope for a certain area. Instead, an area may be specified for the deployment of Mobi-G.KOM, which either covers the whole MANET or only a fraction of it. In the latter case and similar to BlockTree.KOM, multiple instances may be run simultaneously to cover the entire MANET, as already discussed in Section 2.4.2.3.

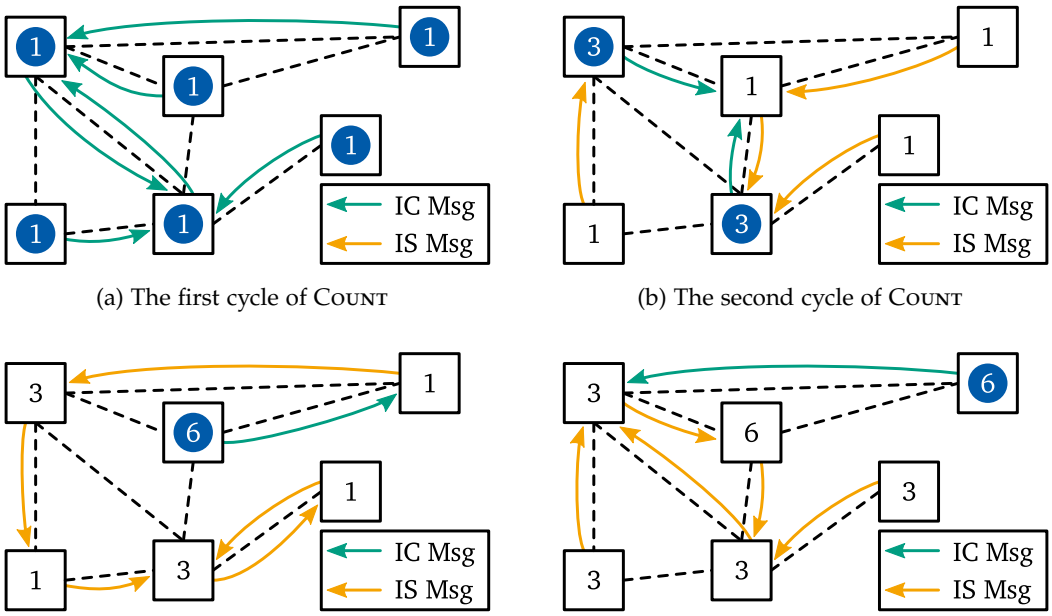
The design and underlying concepts of Mobi-G.KOM fundamentally differ from BlockTree.KOM, as already sketched above and reasoned in the remainder of this chapter. In addition to the implications of the differing design on performance and cost both solutions address the concept of a multi-mechanism for monitoring mechanisms, as outlined during the overview on MAKI in Section 2.2. BlockTree.KOM and Mobi-G.KOM represent two different solutions for monitoring, which may be considered as functionally equivalent. However, they implement the functionality of monitoring in different ways, targeting scenarios with diverging conditions and requirements. According to the basic ideas of MAKI, as outlined in Section 2.2, this diversity may be exploited to execute a transition between the two functionally equivalent mechanisms. This transition facilitates monitoring in communication networks, while the conditions and requirements of the network and the surrounding environment change.

Figure 21 provides an overview on the components of Mobi-G.KOM and outlines the structure of this chapter. We start with a description of the concepts to manage the nodes and organize the monitoring process. These concepts are adopted from the decentralized monitoring mechanism Gossipico from van de Bovenkamp et al. [174], which is originally designed to monitor fixed, large networks of autonomous nodes. On this basis, Section 5.2 deals with the concrete design of Mobi-G.KOM as a decentralized monitoring mechanism for MANETs. The section details the implementation of the three well-known monitoring components, which define (i) the establishment of a monitoring topology, (ii) the collection of monitoring data, and (iii) the dissemination of monitoring results.

5.1 BASICS ON GOSSIPICO

As already indicated in the previous section, Gossipico from van de Bovenkamp et al. [174] does not rely on gossiping to aggregate attributes but to exchange messages. For the processing of the transmitted data Gossipico applies concepts of population algorithms [5], as introduced in Section 3.3.3. In the context of these algorithms a node processes its own as well as the data from another node if both nodes meet. In terms of Gossipico data from both nodes are combined using aggregation. The resulting in-network aggregation of Gossipico complies with the hierarchical computation property. This computation of aggregates exhibits a higher flexibility in contrast to gossip-based approaches, which apply concepts of mass conservation. Contrary to mass conservation, aggregation, which complies with the hierarchical computation property, is not primarily restricted to averaging (cf. Section 3.3.2).

Gossipico divides the time into cycles to coordinate the periodic tasks and the interaction between the nodes. However, it abstains from a synchronization of the cy-



(a) The first cycle of COUNT

(b) The second cycle of COUNT

(c) The third cycle of COUNT

(d) The fourth cycle of COUNT

Figure 22: Example of the first four cycles of the COUNT procedure with the corresponding exchange of Information Collecting (IC) and Information Spreading (IS) messages (adapted from [160])

cles between the nodes by using epochs, as implemented by many other gossip-based monitoring mechanisms [69, 170, 41], which rely on epoch-synced mass conservation for data processing (cf. Section 3.3.2). Gossipico avoids the application of epochs and the related restarts to enable the continuous integration of new values from monitored attributes as detailed at the end of Section 5.1.1. For the data collection and result dissemination Gossipico mainly defines two procedures, which are referred to as COUNT and BEACON. During a cycle, both procedures are executed so that a node must actively start the communication with exactly one of its neighbors per procedure, while several incoming messages may be processed. In the following we start with the description of COUNT, followed by BEACON, and address the steps from van de Bovenkamp et al. to handle the dynamics in fixed networks.

5.1.1 *The COUNT Procedure*

COUNT represents Gossipico's procedure for the collection of data and the dissemination of results, which is sketched in Figure 22. The dashed lines indicate the neighborhood of nodes, which are represented by the squares. A circle inside a square represents a *token* at that node, while the number indicates if the node currently holds one or multiple tokens. The green and yellow arrows indicate the transmission of different types of messages that belong to the COUNT procedure: the green arrow represents an *Information Collecting (IC)* message and the yellow arrow an *Information Spreading (IS)* message. The differences between IC and IS messages are explained in the following.

Message field	Field symbol	Description
Value	V	Denotes the currently monitored and transmitted attributes of the message m
Freshness	F	Specifies the freshness of m
Type	T	Defines the type of m: $T = 0$ represents an IS message, indicating that it disseminates information, whereas $T = 1$ represents an IC message used for data collection.

Table 14: Message format of an IC or IS message

The main idea behind this procedure is that, either in the beginning or if a new node joins the network, every node starts with its own token as depicted in Figure 22a. The token either represents

- the local value of one or several monitored attributes, which have been measured by the node;
- the aggregate of the own token and all tokens from other nodes that have been received in the previous cycle;
- just the aggregated tokens from other nodes that have been received in the previous cycle.

During a cycle, a node sends an IC message with its generated token and hands off its token to one of its neighbors. Every node repeats this operation and sends an IC message per cycle so that the tokens accumulate at a fraction of nodes (cf. Figure 22b) until all tokens are gathered at a single node, as depicted in Figure 22c and 22d. Meanwhile, if a node does not have a token, it sends an IS message to one of its neighbors during a cycle. The IS message indicates how many tokens have been seen by that node during a previous cycle. Newer information that has to be spread takes precedence over older information, ensuring that only recent information is disseminated in the network.

To identify if a message is used to collect or to distribute data as well as to identify the age of data a COUNT message has the format, as listed in Table 14 and as specified by van de Bovenkamp et al. [174]. In this message V specifies a node's current estimate for the global view of an attribute and the age of the sent information is specified by F. Finally, T with $T \in \{0, 1\}$ specifies if the message is an IC message to collect data ($T = 1$) or if it is an IS message to disseminate information ($T = 0$). During a cycle, each node sends its current IC or IS message and subsequently generates an IS message after the transmission of the previous message. This new message, which will be sent during the next cycle, is denoted as waiting message m_w. Until the beginning of the next cycle, the node might receive messages from other nodes, which are denoted as m_r. A received message is processed depending on the type T and freshness F of the waiting and received messages, as specified in Table 15. In this table the subscripts r and w indicate if a received or waiting message with its corresponding message fields is considered. For the case of $T_r = 1$ and $T_w = 1$ the operator $\oplus$ specifies the aggregation of two values of an attribute, where the computation depends on the underlying aggregation function. In addition to the definition

	$T_w = 0$	$T_w = 1$
$T_r = 0$	m_r with higher F_r replaces m_w and updates V_s and V_r with the received values.	m_r is ignored, whereas m_w will be sent.
$T_r = 1$	m_r always replaces m_w.	A new IC message m_w is generated with $V_w = V_r \oplus V_w$, $F_w = F_r + F_w$, and $T_w = 1$. In addition, V_s and V_r are updated with the calculated values.

Table 15: Overview on the processing rules for received messages (abbreviated as m) in Gossipico [174]. The subscripts r and w indicate if the corresponding fields (V, F, and T) belong to the received message m_r or to a waiting message m_w.

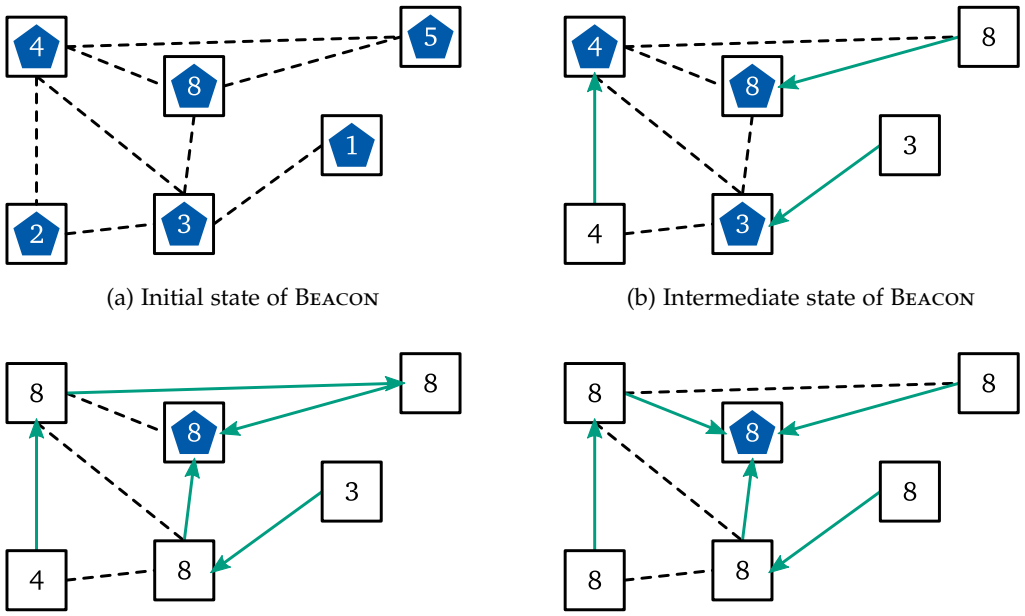
of message fields state values V_s and F_s are introduced, which store the corresponding value V and freshness F at a node. The subscript s indicates that the state values for V and F are considered. A node only updates V_s and F_s if $F_w > F_s$ holds for the waiting message, thus, having a more recent estimate of the global view of an attribute.

To track changes in the local values of monitored attributes and to reflect them in the global view nodes are able to create new tokens. The tokens do not contain the new absolute value of the monitored attribute but the delta to the previously measured value. Due to the continuous adjustment of the monitored attributes Gossipico implements continuous monitoring and does not require a periodic restart to refresh the global view of attributes, as performed by gossip-based approaches [69, 170, 41], which rely on epoch-synced mass conservation for data processing.

5.1.2 *The BEACON Procedure*

The presented COUNT procedure ensures that the data of all nodes, which are distributed over the whole network, are finally collected and processed at a single node. The performance, particularly, the time to collect the data heavily depends on the time it takes that two nodes with tokens meet to exchange their IC messages. To accelerate the data collection and to increase the probability that nodes with tokens meet Gossipico introduces the BEACON procedure. The BEACON procedure represents an autonomous and decentralized election of a leader, which is referred to as *beacon* in the context of Gossipico. By the means of gossiping, the nodes elect the beacon of the network and use the exchanged messages to (i) announce the beacon and (ii) create shortest paths to the beacon. Based on these paths, the tokens are pushed to the beacon and already aggregated and processed along the way, utilizing the IC messages from COUNT.

For the decentralized and autonomous determination of the beacon van de Bovenkamp et al. [174] model this procedure as a fight between multiple armies of different strengths. In the beginning or if a node joins it starts as army leader of its own army. As a leader or member of an army the node competes with nodes from other armies to determine the strongest army with the associated army leader. Figure 23 depicts four different states for the determination of one beacon in the network during the BEACON procedure. The dashed lines highlight again the neighborhood of nodes,



(a) Initial state of BEACON (b) Intermediate state of BEACON

(c) Intermediate state of BEACON (d) Final state of BEACON

Figure 23: Example of different states of the BEACON procedure (adapted from [160])

while a hexagon marks a leader of an army and the number inside quantifies the corresponding strength. The green arrows indicate the predecessor of a node on the path towards the beacon.

At the start of the BEACON procedure and as depicted in Figure 23a, every node belongs to its own army and constitutes the corresponding leader of that army. To determine the beacon a node K stores

- the identifier of its army ID_{army}, where it belongs to;
- the randomly selected strength $S_{ID_{army}}$ of that army;
- the next hop $nh_{ID_{army}}$ of the path to the beacon;
- the corresponding hop count $hc_{ID_{army}}$ to that beacon.

During a cycle and the execution of the COUNT procedure, two neighboring nodes compete with each other and exchange so-called *army messages*. If both nodes belong to the same army, they update the information about the next hop and hop count to the beacon. If they belong to different armies, the stronger army wins and incorporates the defeated node, which (i) becomes a member of that army, (ii) updates its strength, and (iii) stores the winning node as the next hop towards the army leader with the current hop count plus one (cf. Figure 23b and 23c). Figure 23d depicts the final state of the BEACON procedure with one leader of the remaining army. This leader becomes the beacon of the whole network and serves as accelerator for the collection of monitoring data. Table 16 lists and explains again the resulting fields of an army message.

Based on this procedure, an IC message is always sent towards the leader of the current army using the available information at the sending node, whereas IS messages are sent to a random neighbor. As a result, the collected data is sent to the

Message field	Field symbol	Description
Army ID	ID_{army}	Specifies the current ID of an army to distinguish between different armies
Army strength	$S_{ID_{army}}$	Defines the strength of the corresponding army
Next hop	$nh_{ID_{army}}$	Identifies the next hop out of a node's neighborhood on the path towards the current beacon of the army
Hop count	$hc_{ID_{army}}$	Stores the required number of hops to reach the current beacon of the army

Table 16: Message format of an army message

autonomously selected node and already aggregated along the path if possible. For the sake of clarity we will refer to the leader of an army as beacon for the remainder of this thesis.

5.1.3 *Handling Dynamics in Fixed Networks*

To handle the dynamics of the considered networks, which arise from link failures and from the graceful or ungraceful leaving of nodes, the basic assumption of Gossipico is that a node is able to detect a link or node failure in its neighborhood [174]. Once such failure is detected, the affected nodes restart both the COUNT and BEACON procedures by rebuilding their armies and starting a recount. For a successful recreation of armies and recount the following concepts are introduced.

- Recounts work if a node only accepts and processes IC messages from its army and ignores IC messages from other armies. Therefore, if a node becomes a member of a new army, it creates an IC message, includes the locally measured values and sets the freshness to one. The resulting message resembles a message when joining the network.
- To rebuild an army the concepts of *army revival* and *immunity* are introduced. A node, which detects a link or node failure, leaves its current army, recreates its own army and assigns a randomly selected strength to that army. To avoid that the new army is defeated by the previous army and that a recount is suppressed the node's new army is impervious to the previous one. Consequently, the new army conquers the currently reigning army despite its strength to start a successful recount over all nodes.

5.2 MOBI-G.KOM: BRIDGING THE GAP TO MANETS

With MOBI-G.KOM we adapt the basic approach from van de Bovenkamp et al. [174] to be applicable in MANETs, as presented in our previous work [160]. As already indicated in the introduction of this chapter, the goals of the extensions are twofold. On the one hand, we want to create a flat decentralized monitoring mechanism that

does not need to manage a hierarchy in MANETs. On the other hand, we want to design an approach for MANETs that relies on the robust communication pattern of gossiping to exchange but not to aggregate data. In MANETs the utilization of gossiping for data aggregation leads (i) to inaccurate results due to the high network diameter as well as (ii) to mass loss due to message loss.

To design and develop a flat monitoring mechanism for MANETs that fulfills the aforementioned criteria we try to exploit the positive characteristics of MANETs, while counteracting its negative aspects by adapting and extending the initial version of Gossipico where appropriate. In terms of strengths, we exploit the communication medium and its ability to broadcast messages in order to speed up the spreading of information as well as to save bandwidth. Considering weaknesses, we must counteract the missing long-range connectivity accompanied by the resulting high network diameter. Furthermore, the initial design of Gossipico must be extended to handle the highly dynamic environment with constantly changing neighbors. In this regard solutions must be designed (i) to maintain the shortest path to the beacon and (ii) to restart both the COUNT and BEACON procedures for the recount and reactivation of armies. For the remainder of this section we start with the description of adapting the communication between nodes in Section 5.2.1, followed by the required extensions of the COUNT and BEACON procedures in Section 5.2.2 to implement data collection and result dissemination. A description of the establishment of the monitoring topology is omitted, since MOBI-G.KOM does not require any topology to operate on.

5.2.1 *Adapting the Communication*

Similar to BLOCKTREE.KOM MOBI-G.KOM does not rely on an underlying routing protocol to send packets to distant nodes, but implements its own routing methods to communicate. Based on this decision, it operates independently of the routing protocol and may still transmit information if the routing protocol fails, as emphasized by Nanda and Kotz [119]. As detailed in our previous work [160], MOBI-G.KOM tries to make use of broadcasting where possible. The reasons for this decision are twofold.

- With broadcasting, multiple nodes are simultaneously addressable, which helps to reduce the impact of a possibly high network diameter due to the missing long range connectivity.
- Broadcasting preserves the upload bandwidth of a communicating node, because one message is sent to address multiple nodes.

In contrast, broadcasting does not guarantee a successful delivery to all or even a single receiver in the communication range. Thus, broadcasting is particularly of interest for messages, which might be interesting for several nodes, whereas a loss of data is negligible. As a consequence we use broadcasting within the COUNT procedure for the transmission of IS messages for a fast dissemination of information about the global view of monitored attributes in the network. In context of the BEACON procedure, we broadcast army messages for the fast identification of a single beacon in the network and to refresh shortest paths and hop counts to the identified beacon at every node. For both types of messages a loss is not critical, because it is sufficient to obtain the current or even updated information in the next cycle from

the same or a different neighbor, which have been successfully updated during the current cycle. Similar to BLOCKTREE.KOM, we try to make use of the fact that data, which serve as input for the two types of messages, are available at multiple nodes. Consequently, we must not rely on the information exchange between a single source or forwarding node and multiple or, in the worst case, single receivers. There are always sets of potential sources and forwarding nodes, which increase the probability that messages are forwarded.

In contrast, it is necessary that an IC message of COUNT does not get lost. Otherwise, one or several tokens are lost, which leads to an underestimation of the global view of an aggregate. Besides the necessity for a reliable transmission of IC messages, broadcasting of these messages leads to the reception of duplicate information at multiple nodes and to an overestimation of the global view of an aggregate. Contrary to the presented communication concepts in the context of BLOCKTREE.KOM (cf. Section 4.1.2), the exchange of IC messages always takes place between two single nodes and it must be ensured that the transmission of data is successful. For this reason, the transmission of an IC message must be acknowledged by the receiving node. If no acknowledgment is received by the sender of an IC message, it repeats the transmission or pauses the message exchange for the current cycle. The number of retransmissions of an IC message is defined by the system parameter $n_{retrans}$ and the *token acknowledgment timeout* $t_{tokenAckTimeout}$ defines the interval after which a retransmission is started.

5.2.1.1 *Efficient Transmission of new Information*

For the fast and efficient dissemination of IS and army messages in the network, it is mandatory that the messages are not restricted to the neighbors of the source but are forwarded in the whole network. Thus, the initial broadcast of both message types from one or multiple sources is triggered by the cycles In contrast, the subsequent forwarding to the remaining nodes in the network is not synchronized by the cycles but immediately executed after a message has been received. However, if every receiving node immediately broadcasts the received message, the network is congested with IS and army messages. To avoid the resulting broadcast storms [172] and to omit the creation of a consent between potential forwarding candidates, MOBI-G.KOM relies as well on contention-based forwarding schemes, as introduced in [189, 40, 53] and described for the corresponding routing methods of BLOCKTREE.KOM (cf. Section 4.3). Similar to BLOCKTREE.KOM, each node calculates a hesitation time $t_{hesTime}$ based on (7), where $t_{maxForward}$ is a system parameter, which specifies the *maximum forwarding time*, and $k_{hesFactor}$ represents the varying hesitation factor, which is calculated by each node. Due to the fact that the dissemination of IS and army messages may be considered as the dissemination of information in a specified area, $k_{hesFactor}$ is calculated as specified by (13) for the Flooding-Based Area Dissemination in Section 4.3.3.1 and as defined in our previous work [160]. In (13) H^{n+1} represents one of the nodes, which received the broadcasted message from H^n, and r_{max} constitutes a configurable system parameter that specifies the *estimated maximum communication range* of the utilized device. The hesitation factor ranges inside the interval $[0, 1]$ and depends on the distance between the broadcasting node and the receiver, which is normalized by r_{max}. As a result, distant nodes obtain a smaller factor, whereas a higher factor is assigned to neighboring nodes so that distant nodes are favored

to forward a message. Consequently, every IS and army message of MOBI-G.KOM contains the position of the broadcasting node H^n so that a receiving node H^{n+1} is able to calculate its hesitation factor $k_{hesFactor}$.

Similar to BLOCKTREE.KOM, the presented contention-based forwarding scheme avoids that the network is flooded and that the very same message is simultaneously broadcasted by multiple nodes. In contrast to the hierarchical approach, MOBI-G.KOM does not require a message forwarding cache for the identification of redundant messages, which must be dropped. Instead, MOBI-G.KOM evaluates the quality of a received message and checks if the message contains relevant information, which should be forwarded to the remaining nodes. In terms of an IS message, the message contains relevant information if the received data exhibit a higher freshness F_r than F_w of a waiting IS message. An army message is only forwarded, if the sender belongs to a stronger army than the receiver or if a shorter path to a known beacon is discovered.

5.2.1.2 *Responding to Missing or old Information*

Besides the presented procedure for the dissemination of IS and army messages, MOBI-G.KOM additionally specifies a *response procedure*, as introduced in our previous work [160]. The response procedure ensures that a node receives and operates on recent information and does not forward redundant or even old information due to the unreliable broadcasting or to intermittent connections between the nodes. Therefore and as an addition of the previously described qualitative evaluation of a received message, a node checks if a message does not only contain redundant but even old information. Based on the outcome, the broadcasting node of the stale information is informed. The receiver of an old IS or army message broadcasts a corresponding response message with recent information to update the broadcasting sender.

To ensure that the response procedure is triggered every receiver of an old IS or army message becomes a potential candidate to execute the procedure and to increase the probability that the state of the broadcasting node with the old information is refreshed. To avoid that the node is flooded with messages from every node in its communication range the response procedure relies on contention-based forwarding schemes as well. Similar to the previously described calculation of the hesitation factor the distance between the previous sender H^n and a receiver H^{n+1} influences the hesitation time. However, as shown in (20), nearby nodes are favored and obtain a smaller hesitation factor compared to distant nodes. Subsequently, every node calculates its hesitation time $t_{hesTime}$ based on (7). Instead of the system parameter $t_{maxForward}$, which specifies the maximum forwarding time, we rely on a separate system parameter, denoted as *maximum responding time* $t_{maxRespond}$. The maximum responding time represents the constant factor, which is multiplied with $k_{hesFactor}$ to calculate the hesitation time until broadcasting an IS or army message to complete the response procedure. The inverted calculation for the response procedure is used to ensure that the former sender with the stale and inaccurate information receives the updates. The transmission from a nearby node increases the probability that the message is correctly received and that the initial sender H^n does not leave the communication range of the answering node H^{n+1}. Similar to the dissemination of IS

and army messages a hesitating node stops the response procedure if it receives the answer from another node.

$$k_{\text{hesFactor}} = \begin{cases} 1 & \text{if } \left|\overrightarrow{H^{n+1}H^{n}}\right| > r_{\max} \\ \dfrac{\left|\overrightarrow{H^{n+1}H^{n}}\right|}{r_{\max}} & \text{else} \end{cases} \tag{20}$$

5.2.1.3 *Description of the Messages for Data Exchange*

To conclude the section about the adaptation of the communication we present the different message types that are used by MOBI-G.KOM for the data exchange. The description covers the format and size of the message fields and the corresponding data. MOBI-G.KOM contains four different message types. The simplest message is the acknowledgment message, which consists of just one field for the message type and is used to acknowledge the successful transmission of IC messages between two nodes. The remaining three messages consist of multiple fields, which are listed and summarized in Table 17. Since IS and army messages are broadcasted, using the presented contention-based forwarding schemes, both messages require the position of the sender. For the utilization of the current position a node periodically retrieves and refreshes its geographical position. The corresponding interval for the retrieval is specified by the system parameter *position refresh interval* $t_{\text{reqPosInterval}}$. In addition, the IS message comprises the required fields to disseminate the current estimate of an attribute's global view, whereas the army message contains respective information to determine the strongest army with the corresponding beacon. Furthermore, an army message contains the Boolean field *epoch restart*, which indicates if the source of the army message starts a new epoch or just sends an usual army message per information cycle. In Section 5.2.2.2 we describe the utilization of the epoch restart.

The data- and token-field of the IS and IC message are reserved for the transmission of monitoring data, which exhibit the same format. The data comprises the set of aggregates of the monitored attributes, which consists of an attribute's identifier as well as of the fields of the different aggregate functions. For a better comparison with BLOCKTREE.KOM these fields are the same as for BLOCKTREE.KOM and comprise minimum, maximum, sum, and number of nodes. The first three attributes are represented as `double`, whereas the number of nodes is represented as `integer`. In addition to the data- and token-field as well as the freshness `F` to characterize the age of the monitoring data an IC message consists of the IDs of the sender and the message, represented by ID_{node} and ID_{message}. Both fields are used by the receiver of an IC message to identify redundant or old messages, which have been mistakenly retransmitted, because the corresponding acknowledgment did not reach the sender of the IC message.

5.2.2 *Data Collection and Result Dissemination*

After presenting the adaptation of the underlying communication methods to handle the characteristics of MANETs and to support the execution and operation of MOBI-G.KOM, we describe the modifications and extensions to deploy MOBI-G.KOM in MANETs. This section details the design and implementation from two out of the

Message field	Size	Description
Common message fields		
Message type	1 byte	Specifies the message type, which helps to determine how the message and data must be handled
ID_{army}	8 bytes	Holds the ID of the army, the sending node belongs to
IC message fields		
Freshness	4 bytes	Specifies the freshness of the transmitted token
ID_{node}	8 bytes	Identifies the sender of an IC message so that the corresponding receiver is able to identify redundant and old IC messages of that node due to mistakenly sent retransmissions
$ID_{message}$	8 bytes	Identifies an IC message so that a receiver of that message can decide to drop or process the message
Token	Varying	The token represents the collected data, which comprise the aggregates of the monitored attributes. Similar to P- and C-BlockTree.KOM, an aggregate consists of an identifier with a size of one byte as well as of fields to store the minimum, maximum, sum, and number of nodes, which results in an overall size of 29 bytes per monitored attribute.
IS message fields		
Freshness	4 bytes	Specifies the freshness of the disseminated monitoring results, which represent the current estimate of the global view of the monitored attributes
Sender position	16 bytes	Contains the position of the source or sender, which created or forwarded the message
Data	Varying	The data comprise the current estimate of the global view of the monitored attributes. As outlined above, the aggregates of an attribute have a size of 29 bytes.
Army message fields		
$S_{ID_{army}}$	4 bytes	Defines the corresponding strength of the army
Hop count	1 byte	Stores the number of hops of this message towards the beacon of the army
$ID_{impervious}$	8 bytes	Specifies the ID of the army, to which the sending node is impervious
Sender position	16 bytes	Contains the position of the source or sender, which created or forwarded the message
Epoch restart	1 byte	Indicates if the source of the army message starts a new epoch or just sends an usual army message

Table 17: Message format of Mobi-G.KOM's messages

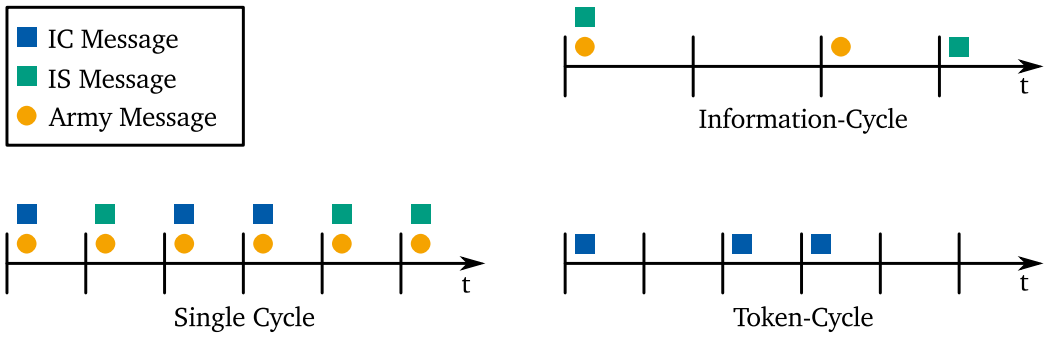
three well-known monitoring components, comprising the collection of monitoring data and the dissemination of the calculated results. The third component, which describes the establishment and management of the underlying topology, is skipped, because no component is required. Since Mobi-G.KOM operates on a flat topology, which corresponds to a mesh and only relies on the neighborhood given by the reachable nodes, the mesh and neighboring nodes must not be managed.

As already outlined in the previous section Count and Beacon are responsible for data collection and result dissemination. Whereas the Beacon procedure is used to identify and manage the beacon of a network and to accelerate the collection of data, the Count procedure is used for the data collection and result dissemination. For the data collection and particularly for the data processing Mobi-G.KOM implements aggregation functions, which comply with the hierarchical computation property. The implemented functions are the same as for BlockTree.KOM and comprise minimum, maximum, sum, number of nodes, and average. To store and transmit this information the corresponding fields of an aggregate consist of minimum, maximum, and sum, which are represented as `double`, whereas the number of nodes is represented as `integer`. The following modifications of both the Count and Beacon procedures are mandatory to handle the characteristics of MANETs, addressing, for instance, the missing long-range connectivity or the high dynamics due to churn and mobility. For this reason

- we modify the clocking cycle, which influences the behavior of a node and triggers the collection and dissemination of data;
- we change the restart procedure for Count and Beacon;
- we actively maintain the different paths to the beacon of an army to guarantee for a good convergence of tokens despite mobility and constantly changing neighbors.

5.2.2.1 *Cycle Adaptation*

As presented before Gossipico utilizes only one cycle to trigger and organize the collection of data, the dissemination of partial or final results, as well as the identification of a single beacon. As depicted in Figure 24a two messages are simultaneously sent during every interval. One of the two messages is used to determine a potentially stronger army and to update the paths to the beacon. The other message either collects data (IC message) or spreads information (IS message) depending on the node's current state and incoming messages. Adapting the length of the cycle influences the sending frequency of all message types. A short interval accelerates the collection process of tokens and helps to react on the missing long-range connectivity and a high network diameter. However, the short interval also influences the transmission of IS and army messages and results in an unnecessary overhead of messages with redundant information. To separate the periodic transmission of IC messages from the remaining two messages Mobi-G.KOM introduces two separate cycles, as already presented in our previous work [160] and depicted in Figure 24b. One cycle is used to trigger the transmission of IC messages. We denote this cycle as *token-cycle* and its interval length is defined by the *token-send-delay* parameter $t_{\text{tokenSendDelay}}$. The second cycle is used for the periodic dissemination of IS and army



(a) Single cycle synchronization of Gossipico (b) Dual cycle synchronization of MOBI-G.KOM

Figure 24: Adaptation of the cycle synchronization in MOBI-G.KOM (adapted from [160])

messages. This cycle is referred to as *information-cycle* and the corresponding interval length is defined by the *refresh timeout* parameter $t_{refreshTimeout}$.

The token-cycle triggers (i) the collection and processing of incoming tokens and (ii) the transmission of IC messages with generated tokens that have been collected during the previous interval. Consequently, a node does not send anything during a token-cycle interval if it does not have a token. As already sketched a short token-cycle leads to a high transmission frequency of IC messages and an accelerated collection of tokens at the beacon, which tackles the problem of a possibly high network diameter. However, a very short token-cycle leads to the immediate forwarding of received tokens without waiting for other incoming tokens. As a result only a few tokens are combined along the path towards the beacon, which leads to the transmission of many tokens that must be handled by the beacon and the nodes next to the beacon. Thus, the corresponding value for the token-send-delay must be carefully selected to provide a fast convergence of tokens at the beacon, while exploiting the aggregation of multiple tokens at intermediate nodes towards the beacon to avoid an increased traffic.

To prevent an unnecessary dissemination of IS and army messages along with IC messages both messages are triggered by the separate information-cycle. Besides the reduction of the sending frequency by a separate cycle, it is not necessary that every node sends a message in an interval, because IS and army messages are broadcasted, using the presented contention-based forwarding scheme. If a hesitating node already receives an IS or army message from another node, it drops its hesitating message if the received information corresponds with the own information. Otherwise, a node has two possibilities to react if the received information differs.

- If the IS or army message contains stale information, the receiving node starts the response procedure to correct the stale entries, as described above. In terms of IS messages, stale information is reflected by the low freshness of results. An army message provides stale information if the node still belongs to a defeated army or maintains a considerably longer path to the beacon. Based on the corresponding calculation of the hesitation time, nearby nodes start the response procedure first to ensure that the initial sender obtains the correct results and that many hesitating nodes in the communication range of the sender receive the message and drop their hesitating messages.

- In terms that the IS or army message contains new information (i.e. higher freshness, stronger army, shorter path to the beacon), the receiving node disseminates the information in the whole network. To accelerate the dissemination and reach as many nodes as possible distant nodes obtain a smaller hesitation time, thus, forward the information earlier.

If MOBI-G.KOM reaches a steady state, where the local information do not differ among the nodes, the transmission of IS and army messages is reduced to a set of nodes, while the majority of nodes drop their hesitating messages.

5.2.2.2 *Counteracting Node Mobility*

Due to the fact that the presented concepts of Gossipico are only targeted at handling the dynamics of fixed networks and are based on assumptions, which do not hold for mobile networks, extensions are necessary to particularly counteract node mobility. For this reason MOBI-G.KOM introduces two additional methods, which prevent that the mobility of nodes disrupts the collection and dissemination of data. The first method addresses the maintenance of the hop count to the beacon at every node and the second one deals with the restart mechanism for the COUNT and BEACON procedure.

MAINTAINING THE HOP COUNT TO THE BEACON

In general, the exchange of army messages is used to maintain and refresh the next hop on the path to the beacon as well as the corresponding hop count. In Gossipico a node always tries to determine the lowest hop count to the beacon to accelerate the collection and dissemination of data. As a result it ignores information about paths with a higher hop count. In terms of MANETs, this approach is not feasible, because nodes are mobile and decrease or, particularly, increase the distance to the beacon. Consequently, transmitted information about a longer path with an increased hop count to the beacon may reflect the effective situation in a MANET and should not be ignored. For this reason and as detailed in our previous work [160] the hop count of a node gradually increases over time if it is not refreshed. As a consequence the node accepts and updates its own state with information about longer paths to the beacon, which reflects the current and effective state in the network.

Besides the influence of mobility on the hop count, it also affects the next hop of the path to the beacon, because the identified node might leave the communication range of the sender of an IC message. In Gossipico this event triggers the restart of the COUNT and BEACON procedure, because it is assumed that a link is broken or a node ungracefully left the network. In a MANET a node just moves away and is not reachable but still online. To avoid that node mobility triggers many and unnecessary restarts MOBI-G.KOM reacts in a different way on disappearing next hops. The acknowledgments of IC messages are used to detect if the next hop has been reached or not. If no acknowledgment of an IC message is received, the node broadcasts an army message with a high hop count to start the response procedure at the receiving nodes. The responding node of the broadcasted army message is used as the next hop for the IC message during the next interval of the token-cycle. Since the calculation of the hesitation time during a response procedure favors nearby nodes, the probability decreases that a next hop leaves the communication range of a sending node before the next interval of the token-cycle.

A TIME-BASED RESTART MECHANISM FOR COUNT AND BEACON

As sketched during the description of counteracting the dynamics in fixed networks (cf. Section 5.1) Gossipico uses an event-based restart mechanism for COUNT and BEACON if a node detects a broken link or a left node. The detection of both events states a problem in MANETs, since nodes might still be online but not reachable, because they left the communication range. As a result the initial event-based restart procedure is not applicable in MANETs, because it leads to unnecessary and many restarts due to the mobility of nodes. As introduced in our previous work [160] MOBI-G.KOM utilizes a *time-based* instead of an event-based restart mechanism. Similar to other gossip-based monitoring mechanisms [69, 170, 41], which rely on epoch-synced mass conservation, MOBI-G.KOM uses epochs to trigger the restart of COUNT and BEACON. After a given amount of time, which is specified by the system parameter *epoch length* $t_{epochLength}$, a node

- enters the new epoch;
- starts as a new beacon of its own army with a randomly selected army strength;
- tries to conquer the MANET to become the only beacon.

To support the restart of COUNT and BEACON every new army is impervious to the formerly reigning army. If a node becomes a member of a new army, it restarts the timer to enter a new epoch, setting the length of the timer to the value of $t_{epochLength}$. Based on this design, no separate message or additional information is required, while only a fraction of nodes, i.e., those nodes where the timer for the new epoch expires, actively triggers the transition into a new epoch. The majority of nodes is automatically passed over into the new epoch.

To prevent that a node loses its global view of the monitored attributes but starts again to collect new tokens after the restart of an epoch MOBI-G.KOM introduces a system parameter, called *intermediate values threshold* $t_{intValThreshold}$. After the node enters a new epoch, it caches the obtained monitoring results from the previous epoch for a certain amount of time, which is specified by $t_{intValThreshold}$. If monitoring results are requested during this interval, a node accesses the cashed instead of the recent results from the current epoch. A disseminated army message contains the flag *epoch restart*, which indicates if the army message effectively starts a new epoch or if it is just an ordinary message, which is triggered by the information-cycle. Based on this flag, nodes do not mistakenly cache the current state but access the recent results.

Besides this extension, every node starts with its own token at the beginning of an epoch, using its locally measured values of the monitored attributes as input. Based on the new values, the global view of a monitored attribute is updated, incorporating the new values of all nodes, which are currently present in the MANET. Due to the epoch-based update of the global view MOBI-G.KOM does not implement continuous monitoring as compared to Gossipico. It abstains from the continuous integration of new tokens that contain the delta of the currently measured value of an attribute to its previous value. Though, if required, a node can also integrate its measured values during an epoch to increase accuracy and freshness of the resulting global view of an attribute. Consequently, MOBI-G.KOM may be configured to implement continuous monitoring as well.

5.3 SUMMARY

MOBI-G.KOM is a flat and decentralized monitoring mechanism for MANETs that operates on a mesh of nodes. It does not require an underlying hierarchy for the identification of communication partners or determination of destinations for data but simply operates on the neighborhood of a node. Consequently, no component is required to establish and maintain a topology. In terms of data collection and result dissemination, MOBI-G.KOM uses gossiping for the exchange of information between the nodes. Contrary to other approaches, which gossip to aggregate data and rely on the concept of mass conservation, MOBI-G.KOM just gossips to communicate and relies on the hierarchical computation property for the processing and aggregation of data. Based on this communication pattern, the approach consists of two separate phases for the data collection and result dissemination: it implements a push-based data collection, which is periodically triggered by the token-cycle, and a proactive dissemination of results, which is triggered by the information-cycle. In Table 18 the corresponding system parameters of MOBI-G.KOM are listed and summarized. The impact of MOBI-G.KOM's different system parameters on performance and cost is evaluated in Section 6.3.3 as well as in Appendix A.3.

System parameter	Symbol	Explanation
Number of IC message retransmissions	$n_{retrans}$	Specifies the maximum number of retransmissions of an IC message
Maximum forwarding time	$t_{maxForward}$	Specifies the constant factor to calculate the hesitation time to forward a message based on (7)
Maximum responding time	$t_{maxRespond}$	Specifies the constant factor to calculate the hesitation time to finish a response operation and transmit an IS or army message
Estimated maximum communication range	r_{max}	Defines the estimated maximum communication range of nodes
Token-send-delay	$t_{tokenSendDelay}$	Specifies the interval length of the token-cycle, which is used for the transmission of IC messages
Refresh timeout	$t_{refreshTimeout}$	Specifies the interval length of the information-cycle, which is used to disseminate IS and army messages
Token acknowledgment timeout	$t_{tokenAckTimeout}$	Defines the interval after which a retransmission is started
Epoch length	$t_{epochLength}$	Defines the duration of an epoch
Position refresh interval	$t_{reqPosInterval}$	Specifies after which amount of time a node retrieves and refreshes the information about its current position
Intermediate values threshold	$t_{intValThreshold}$	Prescribes the duration to cache the obtained monitoring results from a previous epoch, after a new epoch started

Table 18: Overview and description of general system parameters of MOBI-G.KOM

6 MONITORING EVALUATION

After the description of the developed decentralized monitoring mechanisms for MANETs, we conduct a detailed evaluation of Mobi-G.KOM and BlockTree.KOM. Relying on a simulation-based evaluation of both approaches, we investigate if they provide accurate and location-aware monitoring information, while tackling the arising challenges that originate from (i) the dynamic nature of MANETs and (ii) the wireless communication medium with the related devices. In the next section we present the targeted evaluation objectives (cf. Section 6.1) followed by a description of the simulation environment and setup (cf. Section 6.2). The description also covers the considered metrics of the evaluation, which are used to quantify the identified non-functional requirements. After the presentation of the evaluation objectives and setup, we conduct a detailed system parameter evaluation of BlockTree.KOM and Mobi-G.KOM in Section 6.3 and subsequently compare the different monitoring mechanisms with each other (cf. Section 6.4). Finally, Section 6.5 concludes this chapter and summarizes the obtained results.

6.1 EVALUATION OBJECTIVES

As already stated in the beginning of this chapter the major objective is the evaluation of BlockTree.KOM and Mobi-G.KOM to answer the question if both approaches operate in MANETs and provide accurate and location-aware monitoring information, as postulated by the first of the identified research challenges (cf. Section 1.2). In particular, we quantify to which extent the relevant non-functional requirements of a decentralized monitoring mechanism are fulfilled. They are represented by the workload-independent non-functional requirements, comprising accuracy and timeliness in terms of performance as well as cost. To accomplish the overall objective the evaluation is divided into two parts encompassing a detailed system parameter evaluation and a comparative evaluation.

SYSTEM PARAMETER EVALUATION. For a better understanding of BlockTree.KOM and Mobi-G.KOM a detailed system parameter evaluation is conducted, which examines the influence of relevant system parameters on the developed approaches. Based on the evaluation, we define valid parameter ranges and seek for configurations that improve the performance of our developed approaches at minimum or ideally no cost.

COMPARATIVE EVALUATION. The comparative evaluation serves for the comparison of BlockTree.KOM and Mobi-G.KOM. The evaluation consists of a set of experiments with different scenario settings and variations of the corresponding scenario parameters to evaluate and compare both approaches with each other. The scenario parameter variations address scalability and robustness and correspond to the workload-dependent non-functional requirements, as identified in Section 2.4.3. Based on the different setups, we investigate how the

considered approaches handle the second and third research challenge of decentralized monitoring in MANETs and cope with (i) the dynamic nature of MANETs and (ii) the characteristics of the wireless communication medium. As addressed at the beginning of Chapter 5 the comparative evaluation additionally outlines how both monitoring mechanisms perform in the different scenarios. The obtained results may serve as hint to identify potential transitions between the considered monitoring mechanisms to facilitate monitoring in communication networks even if the conditions of the network and surrounding environment change. During the outlook and discussion of the future work in the area of decentralized monitoring (cf. Section 7.2), we elaborate on potential transitions and adaptations based on the obtained results of the following system parameter and comparative evaluation. Furthermore, the comparative evaluation serves for the investigation how the approaches meet the fifth of our identified research challenges and handle the distance-aware relevance of information. Based on this examination, we compare the impact of Mobi-G.KOM's flat topology to BlockTree.KOM's n-tiered tree of blocks.

6.2 SIMULATION ENVIRONMENT

For the evaluation of BlockTree.KOM and Mobi-G.KOM we rely on a simulation-based evaluation for multiple reasons. It allows for studying the large variety of different system scales regarding the spatial size and number of nodes that must be addressed by decentralized monitoring mechanisms in MANETs. Furthermore, it is not bound to the availability of appropriate communication devices when performing a testbed-based evaluation. In addition to the scalability issues simulations enable the isolated examination of specific aspects in an experiment to assess the impact on performance and cost of the mechanisms under test. In our case, simulations facilitate to study the impact of system and especially scenario parameters, for instance, the node density, a node's movement speed, or even characteristics of a monitored attribute. For the specification and, particularly, for the variation of scenario parameters, simulations facilitate the application of synthetic but also realistic workloads. Synthetic workloads may be based on probabilistic models to specify and vary a certain set of scenario parameters. For instance, the applied mobility model of our experiments relies on a probabilistic model to specify the movement speed and destination of a node (cf. Section 6.3.1). Realistic workloads may be derived from traces of real-world measurements so that the resulting specification and variation of a certain set of parameters reflect real-world scenarios. For example, we rely on traces to realistically model the behavior of a monitored attributed in our experiments (cf. Section 6.2.2). Throughout this section, we detail the simulation platform, introduce the modelled attributes, which are monitored by the nodes, and finally describe the evaluation metrics.

6.2.1 *Overview on the Simulation Platform*

For the execution of the evaluation we rely on *PeerfactSim.KOM* [91, 155], which constitutes a simulator that was initially designed for the simulation of P2P systems. The simulator has been extended and additionally enables the simulation of mobile

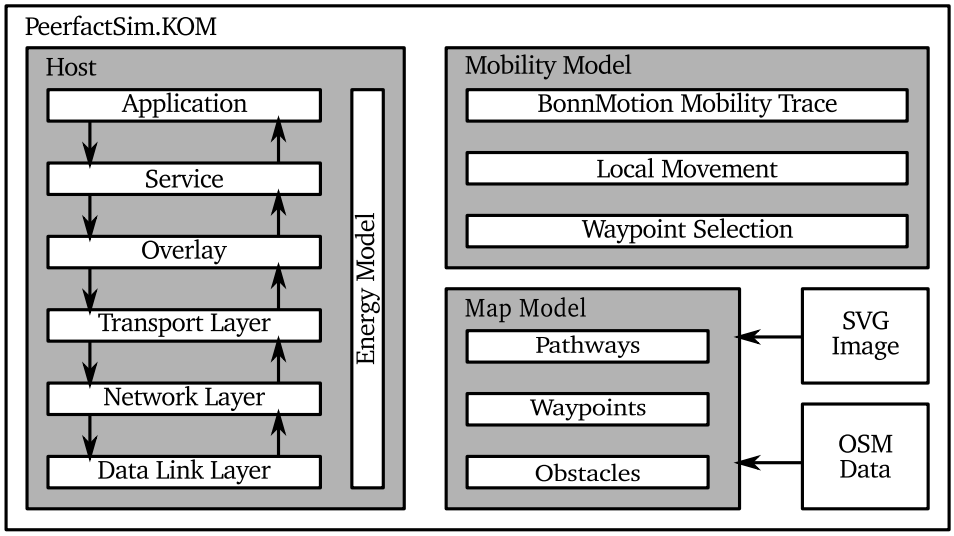


Figure 25: Overview on the simulation platform PeerfactSim.KOM based on [159]

P2P and ad hoc networks [159]. Figure 25 depicts the extended simulation platform that is used in this thesis for the evaluation of BlockTree.KOM and Mobi-G.KOM. The left-hand side of the figure shows the modeled communication layers of the simulator, which construct a host that corresponds to a mobile node in our experiments. Besides the different communication layers, a host in PeerfactSim.KOM may consist of optional models, such as the energy model, which is of interest if mobile devices with limited resources are considered. The right-hand side shows the models, which are used to simulate the mobility of nodes as well as the geographical environment, where the nodes are located. The mobility model can be used with built-in mobility models but enables the integration of generated mobility traces from *BonnMotion* [4]. This tool represents the de-facto standard for the generation of mobility traces based on sophisticated models and is also used in this thesis to model node mobility. The map model simulates the geographical environment and is used to create different settings, for example, with or without obstacles, such as buildings in a city. The creation ranges from purpose-based and simple environments, which are defined by common Scalable Vector Graphics (SVG), to detailed models of urban places or cities based on data from OpenStreetMap (OSM)[1]. In the following we detail the setup and configuration of a host, which represents a participating mobile node in the evaluation. Details on the utilized mobility and geographical environment models are given where required.

6.2.1.1 *Description of the Modelled Communication Layers*

Starting from the bottom of the host, the data link layer comprises a model for the IEEE 802.11g standard [60], which is used to simulate the Wi-Fi ad hoc communication between the devices. The model was ported from the network simulator ns-3 [55] and is used to simulate the communication of mobile communication devices, such as smartphones or tablet PCs. The model assumes a nominal bandwidth of 54 Mbps and an average communication range of 205 m, on which we rely throughout the ex-

1 `http://www.openstreetmap.de` (Last accessed on July 18, 2014)

ecuted experiments. Nodes are able to communicate with a single node (unicast) or multiple nodes (broadcast). On top of the data link layer, the network and transport layer comprise a model for a typical UDP/IP stack that simulates the connectionless communication between two or more nodes. As outlined during the design of BlockTree.KOM and Mobi-G.KOM both approaches avoid an additional overlay layer below the monitoring mechanism but communicate directly with the transport layer. Consequently, the overlay layer of PeerfactSim.KOM is not required for the evaluation, while the service layer of the simulator contains the implemented models of BlockTree.KOM and Mobi-G.KOM, as introduced in Chapter 4 and 5. Finally, the application layer of the simulator is used to provide the decentralized monitoring mechanisms with locally measured values from a set of monitored attributes. Section 6.2.2 details the modeled attributes and reasons their selection.

6.2.1.2 *Description of Additional Components*

Dealing with the additional models of PeerfactSim.KOM for the evaluation, we rely on the energy model to simulate the energy consumption. The reason for considering the influence of a decentralized monitoring mechanism on the energy consumption of a mobile device results from the fact that energy constitutes a primary resource constraint in addition to bandwidth, as outlined by Feeney [38]. Moreover, the energy consumption indicates how the fourth research challenge of decentralized monitoring in MANETs is met (cf. Section 1.2). Consequently, we utilize the energy model of the simulator to measure the resulting energy consumption, arising from communications as well as from the utilization of the GPS receiver.

To measure and simulate the energy consumption we use a dedicated energy model for smartphones, as described in [48]. The model may be classified as a component-based energy consumption model, which is based on measurements from the Nexus One and Milestone 2. The model may also be combined with measurements from recent smartphones, which exhibit other characteristics regarding the energy consumption. However, we assume that the obtained results based on measurements from recent smartphones only differ in their absolute values, whereas the qualitative statements and tendencies will be similar.

Relying on the measurements from the Nexus One, the model assumes that the smartphone has a baseline power consumption of 39 mW, while being in an idle state. If components such as the Wi-Fi chip or the GPS receiver are switched on, the additional power consumption of the respective component is added to the baseline. As the Wi-Fi chip operates in an ad hoc mode the chip is always in an idle state with a power consumption of 314 mW and does not enter a sleep state. If a node sends or receives a packet, the Wi-Fi chip enters a sending or receiving state, which increases the resulting power consumption by about 687 mW or 352 mW for the time to send or receive the packet. Equation (21) describes the energy consumption, while being in the Wi-Fi idle state for the time $t_{\text{Wi-Fi}_{\text{idle}}}$. Equation (22) and (23) describe the energy consumption for sending or receiving a packet for a certain amount of time $t_{\text{Wi-Fi}_{\text{tx}}}$ or $t_{\text{Wi-Fi}_{\text{rx}}}$.

To consider the energy consumption of the GPS receiver we use the model from Zhang et al. [185], which bases on measurements from the Android HTC Dream and HTC Magic. The conducted measurements outline that the signal strength or the number of satellites hardly influence the power consumption. Instead, the power

consumption primarily depends on the state of the GPS receiver, either operating or being in a sleep state. Since we suppose a constant utilization of the GPS receiver for both BlockTree.KOM and Mobi-G.KOM, we assume that the GPS receiver is always active, leading to a constantly increased power consumption of 429.55 mW. Equation (24) describes the energy consumption of an active GPS receiver, while a node is active. Based on the assumption of a component-based energy consumption model as described above, the calculated energy consumption of (24) is added to the calculated energy consumption to communicate, as described by (21) – (23).

$$E(t_{\text{Wi-Fi}_{\text{idle}}}) = (39\,\text{mW} + 314\,\text{mW}) * t_{\text{Wi-Fi}_{\text{idle}}} \tag{21}$$

$$E(t_{\text{Wi-Fi}_{\text{tx}}}) = (39\,\text{mW} + 314\,\text{mW} + 687\,\text{mW}) * t_{\text{Wi-Fi}_{\text{tx}}} \tag{22}$$

$$E(t_{\text{Wi-Fi}_{\text{rx}}}) = (39\,\text{mW} + 314\,\text{mW} + 352\,\text{mW}) * t_{\text{Wi-Fi}_{\text{rx}}} \tag{23}$$

$$E(t_{\text{GPS}_{\text{active}}}) = 429.55\,\text{mW} * t_{\text{GPS}_{\text{active}}} \tag{24}$$

6.2.2 *Modelling Attributes for Monitoring*

To assess the accuracy of a decentralized monitoring mechanism the corresponding attributes must be simulated so that the participating nodes are able to measure and monitor these attributes. The set of the modeled attributes, which are used during this evaluation to examine the accuracy of a monitoring mechanism, consists of three attributes with different characteristics. As detailed and reasoned below the attributes have been selected to evaluate specific aspects of accuracy.

NODE COUNT

Monitoring the number of nodes in a network constitutes a commonly accepted monitoring attribute [156, 158] and a representative problem for collecting network statistics, as outlined by Kostoulas et al. [89]. The obtained results for the number of nodes directly indicate if a decentralized monitoring mechanism captures the complete network or under- or overestimates the actual number of nodes in the network. Moreover, the attribute indicates if every node receives the correct results for the number of nodes in the network. For the evaluation we create a constant attribute that rates every node as one and is used to obtain the number of nodes in the network.

UNIFORMLY DISTRIBUTED SINE FUNCTION

Similar to our previous work [160] the Uniformly Distributed Sine Function (UDSF) represents a synthetic attribute, whose values vary according to a sine function. Graffi [46] identified the sine function as an important attribute, which helps to evaluate how a decentralized monitoring mechanism captures a deterministic yet fluctuating attribute. Contrary to the node count, which provides insights about the completeness of monitoring results, monitoring the sine function reveals how accurate an approach handles an attribute with fluctuating values. Using the time as input parameter, a node computes its local value based on the sine function. To avoid that nodes calculate identical values at the same point in time every node obtains its own sine function, using a uniform distribution, which determines the offset along the y-axis. After determining the local value, the decentralized monitoring mechanism collects these values from the nodes and aggregates them. To rate the accuracy

of monitoring this attribute we rely on the average as aggregation function and compare the average over all monitored values with the average over all effective values. Section 6.2.3 details how the accuracy is calculated for the monitored attributes of a decentralized monitoring mechanism. During the evaluation and similar to our previous work [160], we use a UDSF with a period of one hour.

DA_SENSE MEASUREMENTS

To complete the set of monitored attributes we introduce an attribute, which provides values based on real-world measurements. The measurements are obtained from the participatory sensing application da_sense, which has already been introduced in Section 2.1.2.1 as an example for location-based services. The participatory sensing application da_sense enables users to measure environmental attributes with their smartphone, such as noise level or temperature. Afterwards, users tag the measurements with their current position and upload the location-dependent data to a server, which generates a map with the measured values. We use the measured and location-dependent values for evaluating the performance regarding the provisioning of location-aware monitoring information (cf. Section 6.4.3). In that section, we detail as well how the measured data are processed and integrated into the executed experiments.

6.2.3 *Evaluation Metrics*

To evaluate BLOCKTREE.KOM and MOBI-G.KOM we examine to which extend the workload-independent non-functional requirements are met, while varying the corresponding system parameters as well as parameters of a simulated scenario. As detailed in Section 2.4.3 the workload-independent non-functional requirements comprise accuracy and staleness and characterize the performance of an approach. The occurring communication and energy consumption represent the arising cost. In the following we discuss and select the corresponding metrics, which are used to quantify these non-functional requirements.

ACCURACY

For the quantification of accuracy the error between the monitored and effective global view of an attribute must be calculated. In this context we use the nomenclature from our previous work [159] to denote and calculate the error. With $X_m(A, t, K)$ we denote the *monitored* global aggregate X of a monitoring attribute A that a node K has obtained at time t. The subscript m indicates that the monitored global aggregate is considered. $X_c(A, t, K)$ constitutes the counterpart and the *correct* global aggregate X, as indicated by the subscript c. X_c is obtained by the global knowledge of the simulator and used to calculate the resulting error. Several approaches exist to quantify the error of a decentralized monitoring mechanism comprising the total or relative error. For instance, Kostoulas et al. [89] introduce the *root mean square error (RMSE)* and the *standard deviation of error*, which constitute two orthogonal measures for the quantification of the total error. RMSE quantifies the total difference between the monitored and the real aggregate of an attribute, whereas the standard deviation of error characterizes if the difference often varies or remains constant.

Considine et al. [28] introduce the measure, as specified in (25), to assess the relative error of a decentralized monitoring mechanism. The metric quantifies the relative difference between $X_m(A, t, K)$ and $X_c(A, t, K)$. Throughout the evaluation we rely on the relative error, because it facilitates to rate the obtained results without requiring to know the total values. Taking the node count as an example, a difference of five nodes between the monitored and real value constitutes a poor result if the network consists of 10 nodes, while the accuracy increases if the network consists of 100 or even 1,000 nodes. Moreover and as outlined in our previous work [159], the relative error eases the comparison of results, which are obtained from different approaches in different experiments.

$$\epsilon(X_m(A, t, K), X_c(A, t, K)) = \left| \frac{X_m(A, t, K) - X_c(A, t, K)}{X_c(A, t, K)} \right| \tag{25}$$

Besides the relative error ϵ, we additionally use the same metric but without calculating the absolute value of the difference between $X_m(A, t, K)$ and $X_c(A, t, K)$. We rely on this metric to evaluate (i) how the error is distributed among the nodes and (ii) if the considered approach under- or overestimates the current state of a network.

STALENESS

In terms of staleness we measure $t_{stale} = t_{req} - t_{avg}$ as proposed in our previous works [157, 160]. The metric computes the interval between the point in time t_{req} and the average age t_{avg} of all values that have been considered for the global view. In this equation t_{req} defines when a requested global view of an aggregate is locally available at a node. For example, if an application requires insights on the current state of the network, t_{req} specifies the point in time when the application obtains the information from the monitoring mechanism and can further proceed with the information. For the calculation of t_{avg} we store every point in time when a node adds its locally measured attribute to the aggregate. Finally, the average is calculated over all values before the attribute is disseminated in the network. As a result t_{stale} represents the average staleness of an attribute's global view at that point in time when the global view is requested and received. Dependent on the underlying components to collect and disseminate monitoring information, this request may result in a local lookup at a node, because the data is collected in a push-based manner and results are proactively disseminated. In contrast, a pull-based data collection paired with reactive result dissemination triggers the data collection and result dissemination in the first place.

COST

Based on the classification of Keshav [82], the arising cost of a decentralized monitoring mechanism are divided into three categories: (i) communication, (ii) computation, and (iii) memory cost. Out of the three categories we solely focus on communication cost and evaluate the resulting traffic and power consumption of a node. In terms of the resulting traffic, we measure the upload and download traffic. Dealing with the power consumption, we calculate the resulting power consumption of a node that arises from the transmission and reception of packets and from the utilization of the GPS receiver. We limit our evaluation to communication cost, because the influence on the arising computation and memory cost is negligible: the considered

mobile communication devices are well-equipped with computational power and memory, whereas the presented approaches exhibit a rather small memory footprint and computational overhead.

6.3 SYSTEM PARAMETER EVALUATION

After stating the evaluation objectives and describing the simulation environment, we present the system parameter evaluation of BlockTree.KOM and Mobi-G.KOM. The evaluation consists of multiple experiments per monitoring mechanism, where we vary relevant system parameters and quantify the impact on performance and cost using the metrics described above.

6.3.1 *Description of the Experimental Design*

Before presenting the evaluation results, we introduce the experimental design, which covers the configuration of the default scenario that is used to execute the experiments. For the design of the default scenario we adopt the approach from Kurkowski et al. [95] to define scenarios that meet standards, as described in the following. Based on an exhaustive survey of scenarios for the evaluation of MANET routing protocols they propose a procedure for the definition of reasonable evaluation scenarios. The authors show that the transmission range r of a node is one of the major parameters, which influences the characteristics of a scenario. Examples for these parameters comprise the spatial network size or the movement speed. Consequently, they express distances in relation of the transmission range r. Besides r, the average network partitioning ($\bar{x}_{anp}$) and the average shortest-path hop count ($\bar{x}_{sp}$) constitute further scenario parameters, which essentially influence the characteristics of a scenario and should be used for its definition.

$\bar{x}_{ANP}$ represents the proportion of pairs of nodes without a path during a simulation. The metric characterizes if a small or high amount of network partitioning might occur during a simulation.

$\bar{x}_{SP}$ denotes the average shortest path in the network, which is calculated based on averaging over all existing shortest paths between any two nodes during a simulation. The metric gives insights about how many hops a packet must travel on average to reach its destination.

Based on r, $\bar{x}_{anp}$, and $\bar{x}_{sp}$, Kurkowski et al. [95] calculate the required number of nodes y_{nodes} in (26) as well as the factor y_{area} in (27), which must be multiplied with r^2 to obtain the resulting spatial size of the network.

$$y_{nodes} = e^{-0.164} * \bar{x}_{anp}^{-0.417} * \bar{x}_{sp}^{2.468} \tag{26}$$

$$y_{area} = e^{0.567} * \bar{x}_{anp}^{-0.0769} * \bar{x}_{sp}^{2.159} \tag{27}$$

For the system parameter evaluation we use (26) and (27) to design the scenario. However, we abstain from using both parameters $\bar{x}_{anp}$ and $\bar{x}_{sp}$ but rely on the spatial area size instead of $\bar{x}_{sp}$. The reason for this decision results from the fact that especially C-BlockTree.KOM suffers from an improperly established hierarchy due

to the selection of an inappropriate area. Consequently, we select a square simulation area with an edge length of 1305 m so that C-BlockTree.KOM operates on a completely established hierarchy with three levels ranging from l_0 to l_2. Due to the fact that we want to evaluate if our monitoring mechanisms are capable to monitor the whole network, we reduce the influence of intermittent connectivity between nodes to a minimum and set $\bar{x}_{anp} = 0.1\,\%$. Given the edge length and $\bar{x}_{anp}$, we calculate $\bar{x}_{sp}$ by solving (27) for $\bar{x}_{sp}$ as described by (28). With $r = 205\,m$, $y_{area} = {(1305\,m * 1305\,m)}/{(205\,m * 205\,m)} \approx 40.52$, and $\bar{x}_{anp} = 0.001$ we obtain $\bar{x}_{sp} \approx 3.34$ hops. Subsequently, the number of nodes is calculated based on (26) resulting in 296.71 nodes, which we round up to 297 nodes.

$$\bar{x}_{sp} = e^{\frac{\ln(y_{area})}{2.159}} * e^{-\frac{0.567}{2.159}} * \bar{x}_{anp}^{\frac{0.0769}{2.159}} \tag{28}$$

To model mobility we follow the definitions from Kurkowski et al. and use the steady-state version of the Random Waypoint Mobility model [120], which is commonly used for MANET-related evaluations, as surveyed in [94]. Based on findings of Knoblauch et al. [84] the average movement speed of a pedestrian is 1.51 m/s. Consequently, we set the average movement speed in our evaluation to 1.5 m/s with a minimum of 1 m/s and a maximum of 2 m/s. Pause times for moving nodes are omitted so that every node is constantly moving, which increases the resulting dynamics on the decentralized monitoring mechanism. Finally, to model nodes that leave or enter the area or switch their communication devices on or off, we introduce churn with a mean node session length of 20 min, which corresponds to the time it takes for a node to traverse the map along the diagonal with an average speed of 1.5 m/s. Table 19 summarizes the discussed scenario with the corresponding scenario parameters and gives an overview about how they are configured.

Every experiment with its corresponding scenario is simulated for two hours. During the first hour, the simulation reaches its steady state and the number of active nodes in a simulation levels off at the specified configuration of the scenario parameter. During the second hour, the experiment data is captured from the simulation. For the evaluation of accuracy accompanied by the quantification of the relative error we periodically request and capture the available global view of an attribute from every active node in the MANET. The corresponding interval for the periodic request of monitoring results is set to one minute.

To obtain statistically significant results each experiment is repeated 10 times unless otherwise stated. During the following system parameter and comparative evaluation, we rely on plots with confidence intervals to compare different parameter settings of a particular monitoring mechanism as well as to compare different monitoring mechanisms. The corresponding mean is calculated over one repetition of an experiment, while the plot depicts the average over the 10 calculated means with a confidence of 95%. Besides, we use box plots and cumulative distribution function (CDF) plots to characterize the underlying distribution of results on a per-node basis and to explain the outcome of the plots with confidence intervals. In terms of box plots, the whiskers constitute the 2.5 percentile and 97.5 percentile, the box comprises the results between the first and third quartile, and the line inside the box represents the median.

Scenario parameter	Configuration
Edge length	1305 m
Number of nodes	297
Mobility model	Steady-state Random Waypoint Mobility model [120]
Mean movement speed	1.5 $^{m}/_{s}$
Pause time	0 s
Mean node session length	20 min
Simulation parameter	**Configuration**
Simulation duration	120 min
Measurement duration	60 min
Number of repetitions	10

Table 19: Parameter configuration for the design of the scenario

6.3.2 *System Parameter Evaluation of P- and C-BlockTree.KOM*

This section deals with the evaluation of C- and P-BlockTree.KOM and examines the impact of relevant system parameters on both approaches. In the following we describe the default configurations of C- and P-BlockTree.KOM and identify relevant parameters and parameter combinations for the subsequent evaluation. Given this basis, both approaches are first of all evaluated with and without churn to assess performance and cost in an optimal as well as in an usual scenario (cf. Section 6.3.2.2). The results provide insights on both approaches and outline the differences between them. On this basis we present the system parameter evaluation.

6.3.2.1 *Default Configuration of P- and C-BlockTree.KOM*

The system parameters and their configurations are summarized and listed in Table 20. The table indicates as well, which parameters will be varied in the evaluation. Regarding the fixed system parameters, the first three parameters are used to configure the contention-based forwarding schemes for both approaches. A node uses the estimated maximum communication range to calculate its hesitation factor according to the corresponding forwarding strategy and multiplies the factor with the maximum forwarding time to obtain the final hesitation time. The required information about a node's current position is periodically refreshed every 10 s. Based on the estimated maximum communication range, the block length is defined and set to 145 m, using (5). Regarding the establishment of the hierarchy, we exploit the knowledge about the spatial size of the modeled environment and set the maximum hierarchy level l_{max} for both approaches accordingly. Based on the default configuration of the sector edge length, we set $l_{max} = 2$ for C-BlockTree.KOM and $l_{max} = 4$ for P-BlockTree.KOM.

The remaining parameters belong to the set of system parameters that are varied and evaluated. These parameters are used to configure (i) the underlying hierarchy of

Fixed system parameter	Configuration	Varied system parameter	Configuration
Estimated maximum communication range	205 m	Sector edge length of C-BlockTree.KOM	3
Maximum forwarding time	400 ms	Sector edge length of P-BlockTree.KOM	2
Position refresh interval	10 s	Aggregation interval	15 s
Block length	145 m	Leaf information timeout factor	1.2
Maximum hierarchy level of C-BlockTree.KOM	2	Timeout factor	1.6
Maximum hierarchy level of P-BlockTree.KOM	4	Blocking interval factor	0.9
		Operation blocking interval factor	1.0

Table 20: Overview on the default configuration of the fixed and varied system parameters of P- and C-BlockTree.KOM

P- and C-BlockTree.KOM, (ii) the information exchange over the established hierarchy, and (iii) the caching of data in BlockTree.KOM's tables. If not stated otherwise, the listed parameters are set to their default configuration, as given in the table.

The system parameter evaluation starts with an examination of the sector edge length of P- and C-BlockTree.KOM to separately assess the influence of different topology configurations on performance and cost (cf. Section 6.3.2.3). Subsequently, we evaluate the impact of the remaining parameters on BlockTree.KOM. To avoid that the impact of a potential interdepency between varied parameters is overlooked we rely on a $2^k * r$ factorial experiment design to investigate the effect of individual parameters as well as of parameter combinations, as described by Jain [67]. In the factorial experiment design k represents the number of considered parameters and r states the number of repetitions. After the identification of relevant parameters and combinations, we conduct the detailed evaluation of the remaining system parameters.

For our factorial experiment design we set $k = 3$ and focus on the aggregation interval, timeout factor, and blocking interval factor. The limitation to these parameters results from the fact that (i) the leaf information timeout factor has little impact on BlockTree.KOM and that (ii) the operation blocking interval factor is not suitable for these experiments, because the effect of the system parameter is not unidirectional. We refer the interested reader to Appendix A.2.2 and A.2.6, which detail the evaluation of both parameters.

Table 21 outlines the different configurations, which consist of two variations per parameter and result in $2^3 = 8$ different experiments for every approach of BlockTree.KOM. For a better representation of the results we abbreviate aggregation in-

Configuration	Aggregation interval (AI)	Timeout factor (TF)	Blocking interval factor (BIF)
Lower bound	10 s	1.1	0.6
Upper bound	30 s	3.1	1.2

Table 21: Overview on the different configurations of the three chosen system parameters, which consist of two variations per parameter, representing the lower and upper bound for the configuration range.

System parameter	Proportion of variation [%]					
	Node count	UDSF	Staleness	Upload traffic	Download traffic	Power consumption
AI	3.01	89.09	89.92	75.20	74.85	75.29
TF	34.28	0.81	1.06	0.00	0.00	0.00
BIF	37.81	7.53	6.62	19.47	19.49	19.16
AI ∗ TF	0.02	0.44	0.32	0.00	0.00	0.00
AI ∗ BIF	0.40	1.36	1.39	5.26	5.54	5.45
TF ∗ BIF	24.29	0.34	0.54	0.00	0.00	0.00
AI ∗ TF ∗ BIF	0.01	0.18	0.13	0.00	0.00	0.00
Error	0.17	0.26	0.02	0.07	0.12	0.11

Table 22: Impact of different system parameters of P-BlockTree.KOM on the identified metrics using a $2^k * r$ factorial experiment design

System parameter	Proportion of variation [%]					
	Node count	UDSF	Staleness	Upload traffic	Download traffic	Power consumption
AI	1.58	80.58	84.60	87.57	87.86	88.06
TF	40.63	0.24	3.58	0.02	0.03	0.03
BIF	28.80	14.78	6.56	9.51	8.71	8.62
AI ∗ TF	0.81	1.04	1.38	0.00	0.01	0.01
AI ∗ BIF	0.14	1.92	1.38	2.84	2.74	2.72
TF ∗ BIF	26.27	0.01	1.73	0.01	0.01	0.01
AI ∗ TF ∗ BIF	0.10	0.44	0.52	0.00	0.00	0.00
Error	1.67	1.00	0.25	0.05	0.63	0.56

Table 23: Impact of different system parameters of C-BlockTree.KOM on the identified metrics using a $2^k * r$ factorial experiment design

terval as *AI*, timeout factor as *TF*, and blocking interval factor as *BIF*. Parameter combinations are indicated by the multiplication sign. The results for the impact (in this context also denoted as importance) of the three system parameters and their combinations are listed in Table 22 and 23. As explained in [67] the importance depends on the proportion of variation that is attributable to individual parameters or parameter combinations. The results of both tables confirm that the three individual parameters already play an important role for both approaches and influence performance, cost, or both. Additionally, the experiments reveal that a combination of the timeout and blocking interval factor influences the accuracy of both approaches in terms of the node count.

Based on these insights, we conduct a detailed evaluation of the three individual parameters aggregation interval, timeout factor, and blocking interval factor as well as of the parameter combination of timeout and blocking interval factor. However, in this section, we limit our discussion to the results for the aggregation interval (cf. Section 6.3.2.4). The results for the timeout and blocking interval factor as well as for their combination are presented in Appendix A.2.3, A.2.4, and A.2.5.

6.3.2.2 *Influence of Churn on P- and C-BlockTree.KOM*

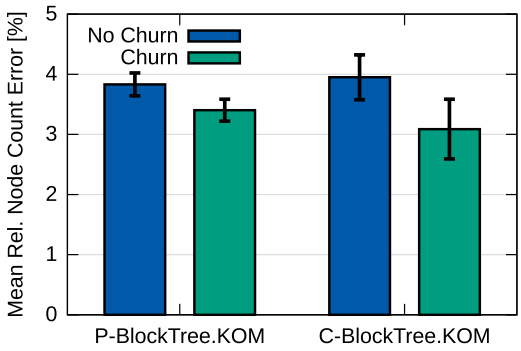
During this section, we examine the influence of the default churn with a mean node session length of 20 min, as listed in Table 19, on P- and C-BlockTree.KOM and outline how both approach handle arriving as well as leaving nodes. Moreover, we use these initial experiments to assess the differences between P- and C-BlockTree.KOM regarding the identified metrics for performance and cost. The obtained results and insights will subsequently serve as reference for the following experiments.

ACCURACY AND STALENESS

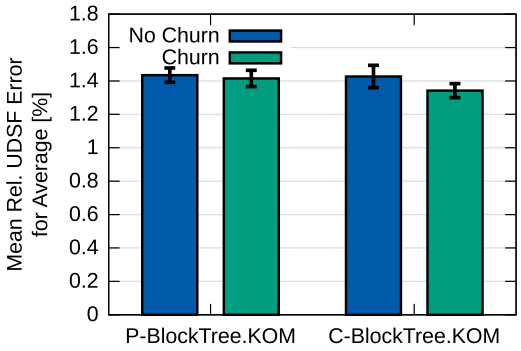
Figure 26 provides an overview about the obtained monitoring results for node count and the Uniformly Distributed Sine Function (UDSF). As specified in Section 6.2.2 the node count attribute is used to capture the currently active nodes in the MANET and the UDSF attribute represents the synthetic attribute, whose values vary according to a sine function.

Figure 26a depicts the mean relative node count error of P- and C-BlockTree.KOM, which reveals the positive impact of churn on both approaches, because the mean relative error significantly decreases. To explain the positive influence of churn on accuracy we examine the distribution of the relative node count error for P- and C-BlockTree.KOM (cf. Figure 26c)[2]. In general it becomes apparent that the error mainly results from an underestimation of the current network state, while only a fraction overestimates the current state. The results of the relative error distribution reveal that the smaller error for churn originates from the reduced underestimation of both approaches, as indicated by the small shift between the dashed and solid curves. For the experiments with churn BlockTree.KOM gathers the monitoring data from nodes, which subsequently leave the network, thus, reducing the effective number of nodes in the network. BlockTree.KOM still incorporates these data into the global view of the attribute, which (i) mitigates the negative impact of underestimation, (ii) reduces the difference between the effective and monitored number of nodes in the network, and (iii) leads to a reduced error.

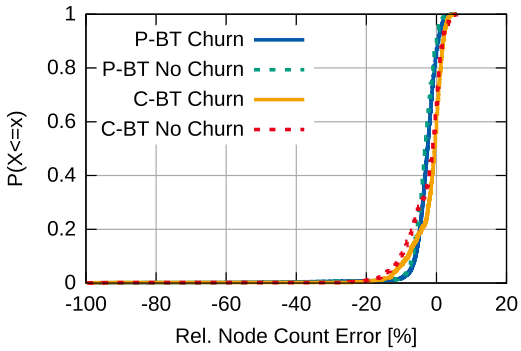
2 In Figure 26c and 26d, P-BlockTree.KOM is abbreviated as *P-BT* and C-BlockTree.KOM as *C-BT*.



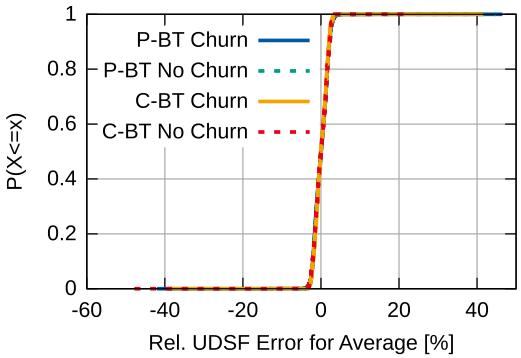
(a) Mean relative node count error



(b) Mean relative UDSF error using the average as aggregation function



(c) Cumulative distribution function of the relative node count error



(d) Cumulative distribution function of the relative UDSF error

Figure 26: Overview on accuracy of the provided monitoring results for node count and UDFS

Regarding the mean relative error for UDSF, the results in Figure 26b provide two insights. On the one hand, P- and C-BlockTree.KOM accurately monitor fluctuating attributes with a mean relative error of 1.43 % and 1.42 % without and with churn in terms of P-BlockTree.KOM and of 1.43 % and 1.34 % without and with churn in terms of C-BlockTree.KOM. On the other hand, the results outline the small impact of churn on the accuracy of this attribute, because no considerable change is observable for P- and C-BlockTree.KOM. The reasons for the accurate monitoring of that attribute and the slight impact of churn stem from two aspects. In general the continuous updates of attributes as well as the staggered collection and dissemination of data accelerate the integration of new values into the global view of an attribute. Furthermore, missing or new values of UDSF due to churn hardly influence the monitored global view, especially when the average is applied as aggregation function. Consequently, the distribution of the relative UDSF error in Figure 26d reveals for P-BlockTree.KOM that 95 % of the monitoring results exhibit an error from the interval [-2.74, 2.94] and [-2.73, 2.92] without and with churn. Regarding C-BlockTree.KOM, the error ranges in the interval [-2.78, 2.83] and [-2.66, 2.79], respectively.

To this moment, the results for accuracy do not exhibit noticeable differences between P- and C-BlockTree.KOM, as the mean relative error does not significantly differ between both approaches, when neglecting the mean relative UDSF error for C-BlockTree.KOM with churn. This constitutes a remarkable outcome, because P-

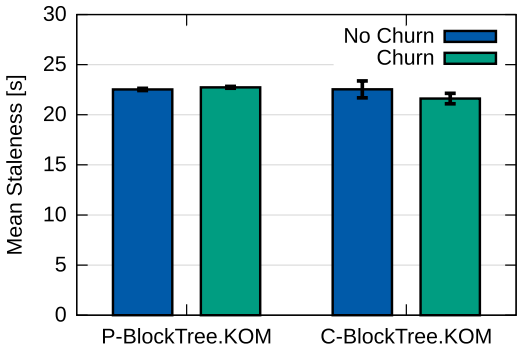


Figure 27: Mean staleness of the provided monitoring results

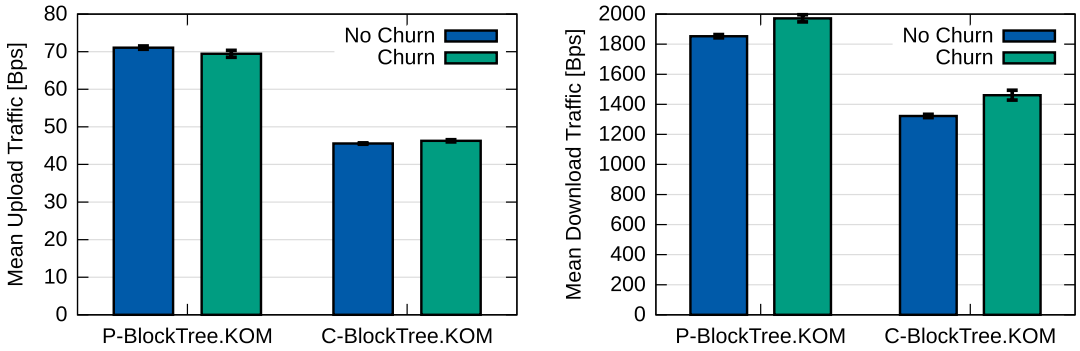
BlockTree.KOM requires five levels (from l_0 to l_4) to cover the whole area in the scenario. However, the staggered execution of Aggregating-Up operations at the different levels compensates the higher hierarchy. In contrast, C-BlockTree.KOM already covers the whole area with a hierarchy of three levels (from l_0 to l_2). Nevertheless, the data collection is slower than for P-BlockTree.KOM, because the Aggregating-Up operation is only triggered once per aggregation interval.

In terms of staleness, similar tendencies are observable for (i) the influence of churn and (ii) the differences between P- and C-BlockTree.KOM, as depicted in Figure 27. Dealing with the differences, the freshness of the monitoring results is comparable between both approaches, although the maximum level differs between the two hierarchies. As previously discussed, P-BlockTree.KOM compensates the higher level by the staggered execution of the Aggregating-Up operation per level. Considering the influence of churn, the results reveal that arriving and leaving nodes have a minor impact on the staleness of results. To conclude, both approaches of BlockTree.KOM provide accurate and timely results and are not severely affected by the increasing dynamics through arriving and leaving nodes.

TRAFFIC AND POWER CONSUMPTION

Regarding cost, Figure 28 provides an overview on the resulting overall upload and download traffic of both approaches. Starting again with the comparison between P- and C-BlockTree.KOM, Figure 28a and 28b reveal that the mean upload and download traffic of P-BlockTree.KOM are considerably higher than of C-BlockTree.KOM. The staggered execution of Aggregating-Up operations per level during an aggregation interval leads to a fast data collection with implicit result dissemination but comes at the expense of an increased mean upload and download traffic, as detailed in the following. A node of P-BlockTree.KOM belongs to every active level in the established hierarchy. Consequently, it executes an Aggregating-Up operation per level and must forward the corresponding information per level, whereas a node of C-BlockTree.KOM executes this operation only for the next higher level in the hierarchy. In C-BlockTree.KOM, a node and its surrounding block belong to exactly one level in the hierarchy and the data must not be spread to the respective dissemination area but has to be sent to another block.

Assessing the influence of churn on the resulting traffic of both approaches, an interesting effect is observed, which becomes specifically apparent by considering the resulting upload and download traffic of P-BlockTree.KOM. The overall upload traffic slightly but significantly decreases, whereas the resulting download traffic in-



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 28: Overview on the overall mean traffic per node

creases. The reason for this effect arises from the node density in combination with the contention-based forwarding scheme for the transmission of AGGREGATING-UP messages. Due to the application of churn within the scenario the current number of active nodes is most of the time above the specified configuration of 297 nodes, which leads to a slightly increased node density. The increased density results in a higher ratio between potential candidates for a transmission and effective sending nodes, which reduces the mean upload traffic per node. In contrast, a higher density increases the probability that the nodes receive new or redundant AGGREGATING-UP messages, which leads to the augmented mean download traffic. Since the AGGREGATING-UP traffic mainly influences P-BLOCKTREE.KOM's traffic, the effect is reflected by the decreasing upload traffic and the increasing download traffic (cf. Figure 28a and 28b). During the comparative evaluation where we examine the impact of node density on the developed approaches and strongly vary the node density this effect will become more obvious and even observable for C-BLOCKTREE.KOM. A detailed overview on the arising upload and download traffic of P- and C-BLOCKTREE.KOM's message types, underpinning the statements above, can be found in Appendix A.2.1.

Finally, the arising cost in terms of the mean power consumption are depicted in Figure 29. The results do not represent the overall mean power consumption but the supplementary consumption, which arises from the transmission and reception of messages. If a node does neither send nor receive a message, it still consumes 782.55 mW for its active GPS receiver and the idle state of the Wi-Fi chip, as described in Section 6.2.1.2. In general, the results reveal that the mean power consumption mainly depends on the mean download traffic per node, which accounts for a large fraction of the overall traffic per node. Consequently, the plots in Figure 29 resemble the results for the mean download traffic in Figure 28b. Thus, P-BLOCKTREE.KOM exhibits a higher mean power consumption than C-BLOCKTREE.KOM, because more data are sent and received by the planar approach. Taking the impact of churn into consideration, the mean energy consumption increases for both approaches, similar to the results for the mean download traffic.

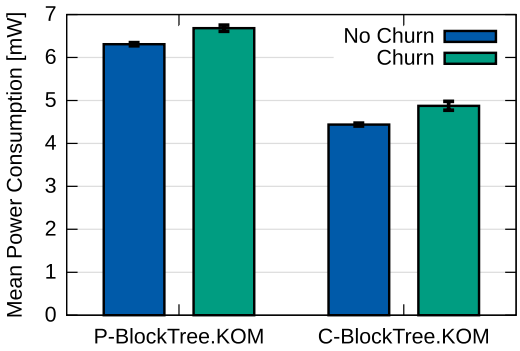


Figure 29: Mean power consumption per node

6.3.2.3 *Variation of the Hierarchy*

During this section, we vary the underlying hierarchy of both approaches. In terms of P-BlockTree.KOM, we alter the number of federated sectors that represent a single sector at the next higher level in the hierarchy. Table 24 lists the different configurations for the sector edge length parameter, which specifies how many sectors are federated along the edge of a sector at the next level. Moreover, the table depicts the impact of more sectors per level on the resulting height of the hierarchy, because the level decreases with more federated sectors along an edge.

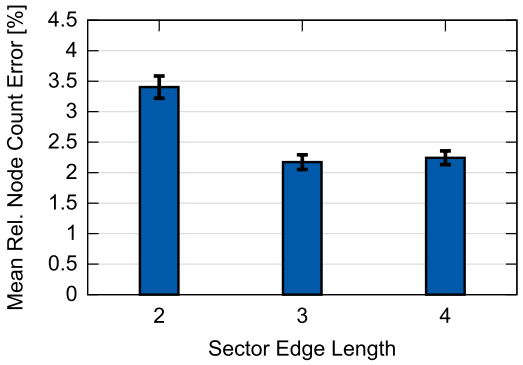
System parameter	**Configuration**
Sector edge length	2, 3, 4
Maximum level	4, 2, 2

Table 24: System parameter variation of P-BlockTree.KOM's sector edge length. The underlined value represents the default configuration of the system parameter.

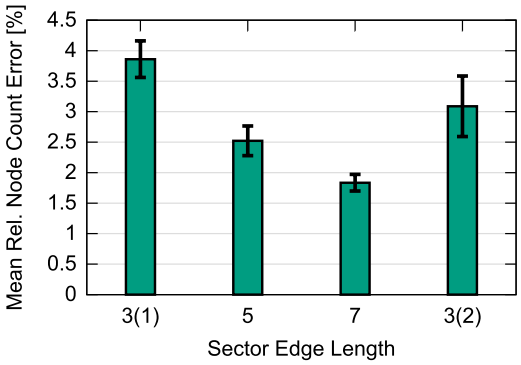
System parameter	**Configuration**
Sector edge length	3(1), 5, 7, 3(2)
Edge length	435, 725, 1015, 1305
Number of nodes	33, 92, 180, 297
Maximum level	1, 1, 1, 2

Table 25: System parameter variation of C-BlockTree.KOM's sector edge length. The underlined value represents the default configuration of the system parameter.

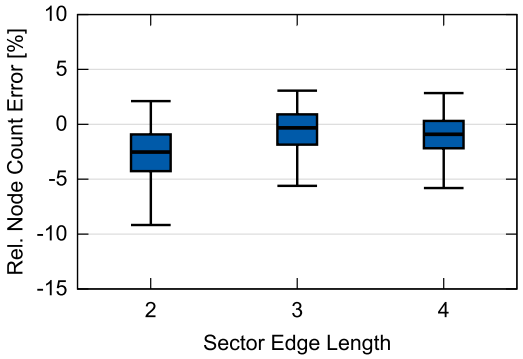
The different configurations for the particular parameter of C-BlockTree.KOM are depicted in Table 25. Unfortunately, a variation of the length of a sector changes the covered area of the monitoring mechanism. To guarantee a proper establishment of C-BlockTree.KOM's hierarchy for the exclusive assessment of the impact of differently configured topologies we must vary the spatial network size and the number of nodes as well. The experiments, comprising (i) sector edge length, (ii) the new spatial network size (expressed by the edge length), (iii) the number of nodes, and (iv) the resulting height of the hierarchy, are listed in Table 25. To complete the experiments for C-BlockTree.KOM a fourth experiment is added with a sector edge length of three but a maximum level of one, which serves for an additional comparison with



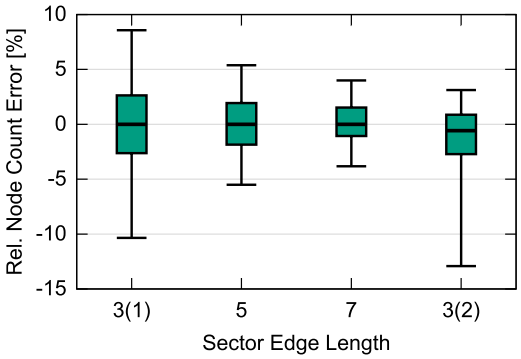
(a) Mean relative node count error of P-BLOCKTREE.KOM



(b) Mean relative node count error of C-BLOCKTREE.KOM



(c) Distribution of the relative node count error of all collected measurements from P-BLOCKTREE.KOM



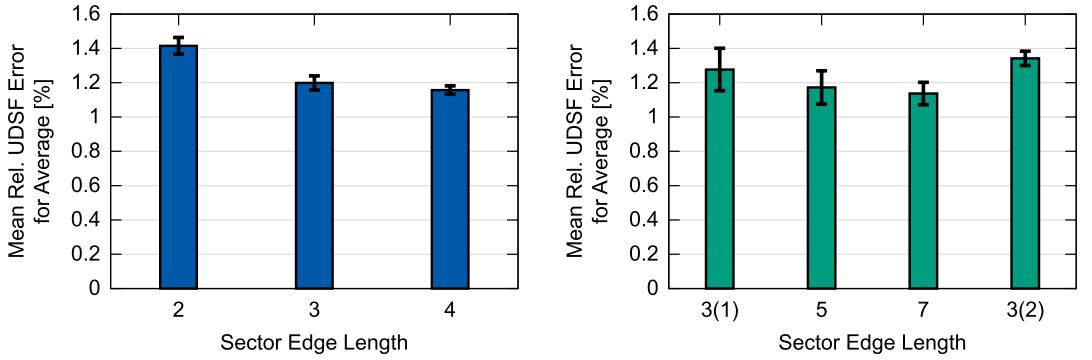
(d) Distribution of the relative node count error of all collected measurements from C-BLOCKTREE.KOM

Figure 30: Overview on the mean relative node count error for different configurations of the hierarchy

the other two experiments that have a maximum level of one. We distinguish between the two experiments with a sector edge length of three by adding the resulting level in brackets, as listed in Table 25.

ACCURACY AND STALENESS

Figure 30a depicts the influence of P-BLOCKTREE.KOM's varied sector edge length on the mean relative node count error. The results uncover that more than two federated sectors along a sector edge significantly reduce the mean relative error, because the latter two configurations require only two levels to establish the hierarchy. Consequently, the difference between an edge length of three and four hardly differs. The positive impact of a reduced number of levels becomes also apparent by two other observations. First, the mean staleness for a sector edge length of three and four decreases (cf. Figure 32a), because only two levels must be passed to reach the highest level of the hierarchy. As underpinned by the results this accelerates the collection and dissemination of data. Second, the collected monitoring data are only cached twice in the hierarchy table at intermediate levels so that older data are not prematurely purged from these tables by the rather aggressively configured timeout factor. The distribution of the relative node count error in Figure 30c underpins the latter observation. The illustrated results reveal that the larger mean error for a sector edge



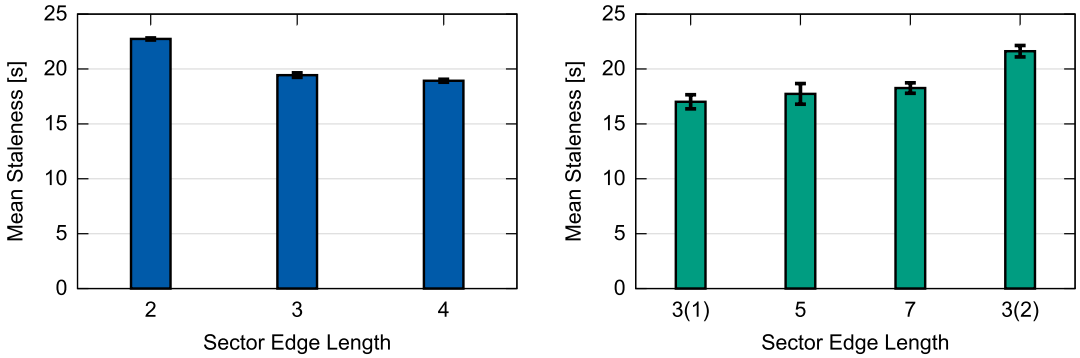
(a) Mean relative UDSF error of P-BlockTree.KOM using the average as aggregation function

(b) Mean relative UDSF error of C-BlockTree.KOM using the average as aggregation function

Figure 31: Overview on the mean relative UDSF error for different configurations of the hierarchy

length of two originates from an increased underestimation, which is attributed to the premature deletion of data from the hierarchy table at four instead of two levels. Dealing with mean relative UDSF error, Figure 31a shows that a flat hierarchy is beneficial for monitoring fluctuating attributes. The accelerated collection and dissemination of data reduces the difference between the monitored and effective global state of the attribute. Consequently, the mean result staleness and the mean relative UDSF error are reduced.

Regarding C-BlockTree.KOM, Figure 30b depicts the mean relative node count error. The results for the first three experiments with $l_{max} = 1$ reveal that a longer edge length of the sector has a positive effect on accuracy, because the mean relative node count error decreases, despite the increasing spatial network size and number of nodes. The reduction of the mean relative error is attributed to the increased spatial network size and number of nodes, which reduce the fraction of under- as well as overestimating nodes, as depicted by the shrinking whiskers and boxes in Figure 30d. Dealing with our default configuration, the accuracy increases in contrast to the configuration with the same sector edge length but a hierarchy with a maximum level of one. Figure 30d uncovers that the reduction of the mean error particularly originates from a reduced overestimation of the current state. However, the default configuration cannot compete with the achieved accuracy of the remaining two experiments due to an increasing underestimation, which originates from the additional caching and premature deletion of data at the second level in the hierarchy table. As far as the mean relative UDSF error is considered the results in Figure 31b reveal a positive influence of the increasing sector edge length on accuracy. In contrast the accuracy decreases for the default configuration. The increasing mean relative error for the default configuration is explained by taking the staleness of monitoring results into consideration. Figure 32b outlines that the result staleness increases for the default configuration due to the adding of another hierarchy level. Wheras a decreasing staleness improves the accuracy of monitoring a fluctuating attribute, as discussed above, the increasing staleness has the opposed effect and decreases the accuracy of monitoring this attribute.



(a) Mean staleness of the provided monitoring results of P-BlockTree.KOM

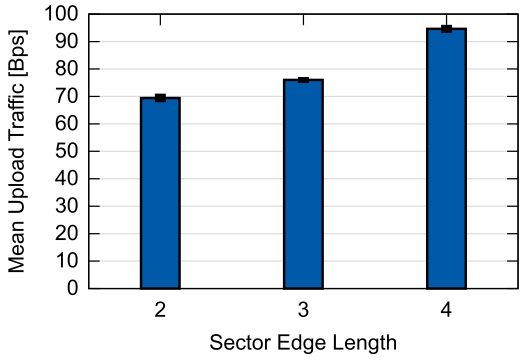
(b) Mean staleness of the provided monitoring results of C-BlockTree.KOM

Figure 32: Overview on the impact of the hierarchy on the freshness of monitoring results

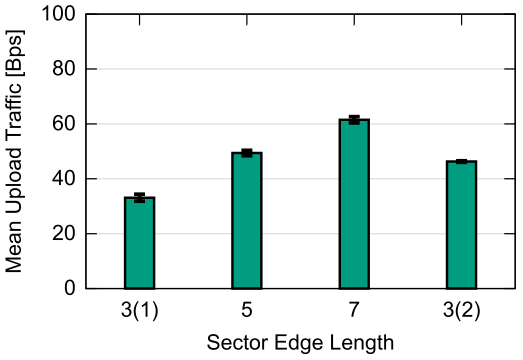
TRAFFIC

The influence of a varying hierarchy on the arising cost in terms of the overall upload and download traffic is depicted in Figure 33. Starting again with the influence on P-BlockTree.KOM, Figure 33a and 33c reveal that the traffic grows as a function of the sector edge length. The observation leads to the conclusion that the previously observed gain in performance between the default and remaining two configurations of the sector edge length comes at increased cost. Even worse, the upload and download traffic increase for a sector edge length of four, while the performance in terms of accuracy and timeliness remains nearly constant, as compared to a sector edge length of three. In addition the results uncover that the associated reduction of hierarchy levels does not lead to reduced cost, although, the effective execution of Aggregating-Up operations decreases due to the reduced number of levels. To understand this phenomenon we focus on the download traffic that arises from the executed Aggregating-Up operations and accounts for a large part of the total download traffic (cf. Figure 34a). The increased mean Aggregating-Up traffic per node can be explained when considering the highest level of P-BlockTree.KOM's hierarchy, which either consists of 2^2, 3^2, or 4^2 sectors. Every sector must send its information to the remaining sectors and, in turn, receives their information. Using the contention-based forwarding scheme paired with the message forwarding cache, a delayed message from a certain sector is dropped if a message from the same sector has been previously received. With fewer sectors the probability increases that this event occurs so that the received message is dropped and must not be disseminated in the network. If this message originates from the same node but of another sector, the message must still be forwarded. These circumstances lead to the increased total traffic, because fewer levels do not compensate a higher amount of sectors per level with the associated dissemination of information among them.

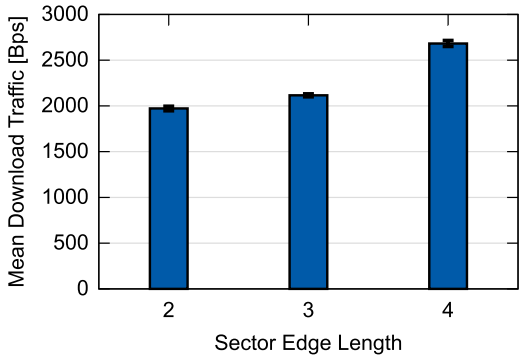
The depicted results of C-BlockTree.KOM in Figure 33b and 33d provide two main findings: a longer edge length of the sector as well as the extension of the hierarchy with an additional level lead to an increased overall upload and download traffic. We focus on the latter aspect in more detail during the comparative evaluation in Section 6.4.1.1 when we investigate the influence of an increasing spatial network size on the developed decentralized monitoring mechanisms. During the remainder of this section, we investigate the causes for the increased upload and



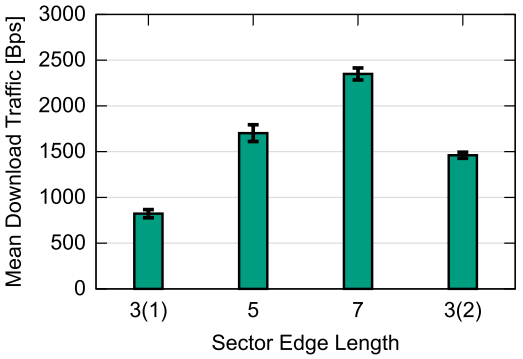
(a) Mean overall upload traffic per node of P-BLOCKTREE.KOM



(b) Mean overall upload traffic per node of C-BLOCKTREE.KOM



(c) Mean overall download traffic per node of P-BLOCKTREE.KOM

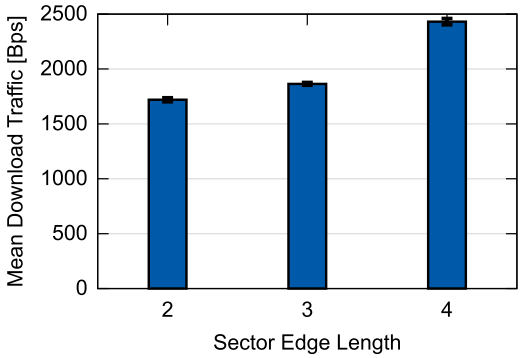


(d) Mean overall download traffic per node of C-BLOCKTREE.KOM

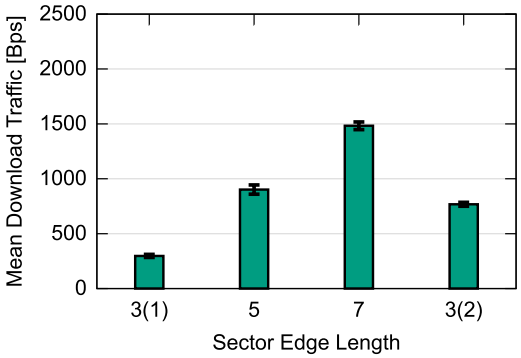
Figure 33: Overview on the overall mean traffic per node for different configurations of the hierarchy

download traffic in dependence of the growing sector edge length by looking at C-BLOCKTREE.KOM's different traffic types.

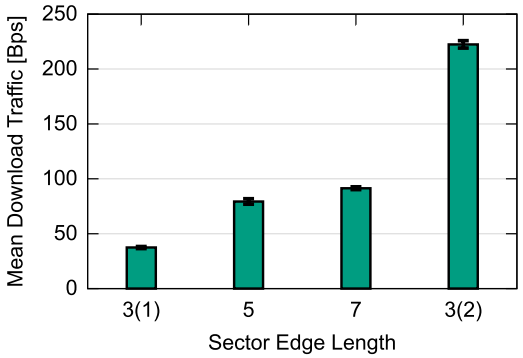
Figure 34b, 34c, and 34d depict the mean download traffic per second per node for the AGGREGATING-UP, DISSEMINATING-DOWN, and replication operations, respectively. The results already reveal that the growing overall traffic mainly originates from the AGGREGATING-UP traffic, which accounts for a large fraction of the overall traffic. In terms of a sector length of five or seven, the concentrating block must handle the information from more than eight surrounding blocks. Furthermore, many blocks must forward the data from other blocks in addition to their own data so that the information is concentrated at the responsible block. Both factors lead to the depicted effect of an increased AGGREGATING-UP traffic. The fourth experiment with a sector length of three and a maximum hierarchy level of two already benefits from the hierarchical topology. Due to the partitioning into multiple sectors at different levels every concentrating block is only responsible for its surrounding eight blocks and eight sectors at the second level. This partitioning is reflected by the decreased AGGREGATING-UP traffic, compared to the two experiments with a sector length of five and seven. The positive effect of a smaller sector becomes also apparent by the depicted replication traffic in Figure 34d. With a shorter length of the sector concentrating blocks must replicate the data from a reduced number of surrounding blocks, which is reflected by the decreased mean download traffic. In terms of the fourth



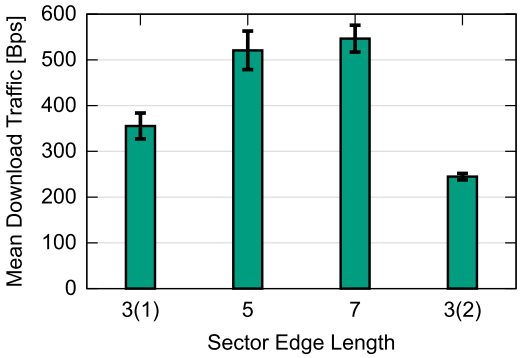
(a) Mean download traffic per node of P-Block-Tree.KOM arising from the execution of Aggregating-Up operations



(b) Mean download traffic per node of C-Block-Tree.KOM arising from the execution of Aggregating-Up operations



(c) Mean download traffic per node of C-Block-Tree.KOM arising from the execution of Disseminating-Down operations



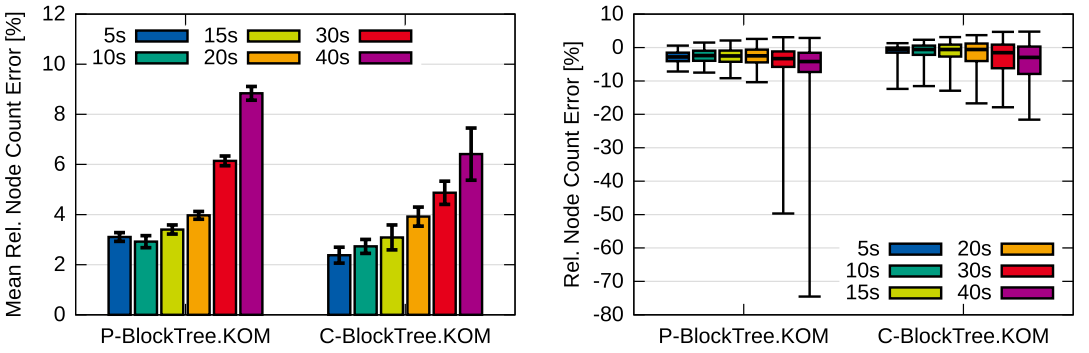
(d) Mean download traffic per node of C-Block-Tree.KOM arising from the replication of information in concentrating blocks

Figure 34: Overview on the download traffic per node for the transmission of different messages for different configurations of the hierarchy

experiment, the small replication traffic per node is explained as follows. In total, this experiment generates the most replication traffic compared to the remaining experiments. However, due to the fact (i) that the traffic is distributed among a larger number of nodes and (ii) that the replicated data only comprise information from eight instead of 24 or even 48 blocks, the effective mean replication traffic per node decreases.

Whereas an additional level reduces the arising Aggregating-Up and replication traffic per node, Figure 34c reveals that it increases the Disseminating-Down traffic. For the first three experiments every node just has to forward the monitoring results from one concentrating block. In terms of the last experiment with a maximum level of two, every node must forward the monitoring results from two concentrating blocks. However, the increased Disseminating-Down traffic does not counterbalance the increasing replication and Aggregating-Up traffic of the other experiments. Consequently, the second and third experiments lead to an increased mean total traffic per node.

We omit the results for the mean power consumption per node, because they only reflect the tendencies from the discussion about the overall mean upload and download traffic.



(a) Mean relative node count error

(b) Distribution of the relative node count error of all collected measurements

Figure 35: Overview on the mean relative node count error for a varying aggregation interval

6.3.2.4 *Aggregation Interval*

After assessing the impact of different configurations of the hierarchy on both approaches, we continue with the evaluation of the aggregation interval. Apart from the parameters for the configuration of the topology, the corresponding aggregation interval parameter constitutes the most important parameter for the information exchange, because it defines the interval to trigger the periodic operations. Table 26 lists the different configurations for the experiments, where an interval of 15 s constitutes the default configuration. During the evaluation, we study the impact of shorter as well as longer intervals to examine the cost for a possible gain but also reduction in performance. For this reason we extend the upper and lower bound of the configuration range, as introduced in Section 6.3.2.1, by a configuration of five seconds and 40 seconds to stress the impact of a shorter and longer aggregation interval.

System parameter	**Configuration**
Aggregation interval [s]	5, 10, <u>15</u>, 20, 30, 40

Table 26: System parameter variation of the aggregation interval. The underlined value represents the default configuration of the system parameter.

ACCURACY AND STALENESS

Figure 35 depicts the results for the relative node count error, comprising the mean relative error (cf. Figure 35a) and the relative error distribution of all collected measurements (cf. Figure 35b). Given the default configuration of 15 s, the results reveal that a shorter aggregation interval just slightly improves the accuracy of both approaches with a mean relative error that levels out above two percent. In terms of P-BlockTree.KOM, the result for five seconds even indicates that the mean relative error increases. Taking a look at the relative error distribution in Figure 35b, it becomes apparent that the increasing error for this configuration arises from a growing underestimation, because even a shorter interval does not compensate the data loss during the collection over the established hierarchy. In addition the distribution uncovers that P-BlockTree.KOM generally suffers throughout all experiments from an underestimation. For a longer aggregation interval, the mean relative error signifi-

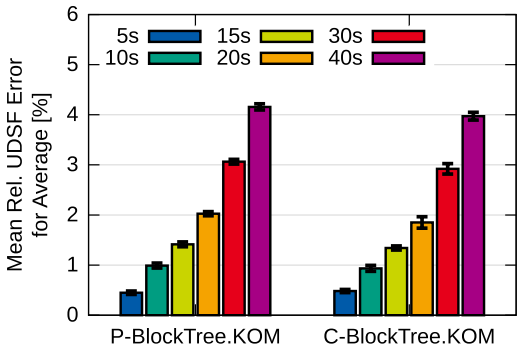


Figure 36: Mean relative UDSF error for a varying aggregation interval

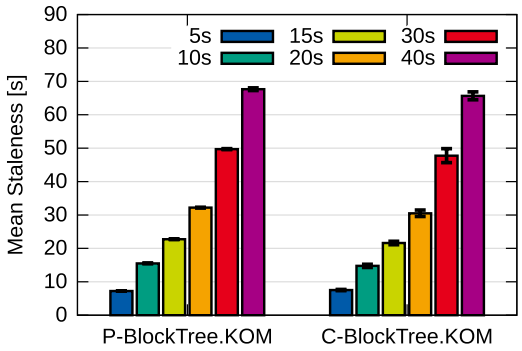


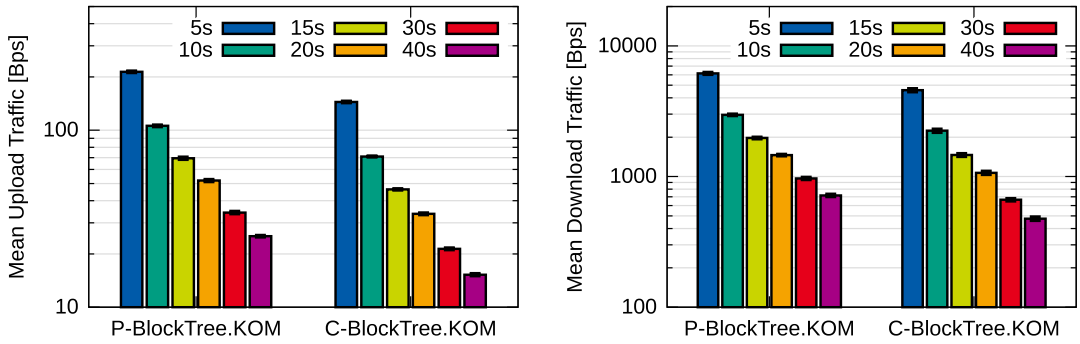
Figure 37: Mean result staleness for a varying aggregation interval

cantly increases for both approaches. However, the longer aggregation interval has a stronger influence on P- than on C-BlockTree.KOM, as the latter approach collects and disseminates the data only over three instead of five levels. The relative error distribution reveals that P-BlockTree.KOM suffers from a growing fraction of outliers, as indicated by the extended lower whisker. Regarding C-BlockTree.KOM, the increasing error is primarily attributed to the growing number of nodes that underestimates the current state for longer aggregation intervals, as illustrated by the growing and sinking box.

The mean relative UDSF error, which is depicted in Figure 36, reveals that the aggregation interval influences the monitoring accuracy of this attribute in both directions. Given again the default configuration of 15 s, extending as well as shortening the duration leads to an increased but also decreased mean relative error, contrary to the results for node count, where the influence of a shortened interval mitigates. In fact, it becomes apparent for both approaches that doubling the length of the interval doubles the error, as reflected by the outcome for five seconds, 10 s, 20 s, and 40 s. In contrast to the relative node count error lost or missing data do not influence the monitoring accuracy of UDSF that strong, because the average is used as aggregation function. Instead, the accuracy heavily depends on the time it takes to collect and disseminate the data. This time corresponds to the mean staleness, which increases with a longer aggregation interval, as depicted in Figure 37, and, thus, explains the growing mean relative UDSF error. Focusing on the differences between P- and C-BlockTree.KOM in terms of UDSF, the results in Figure 36 and 37 uncover that the accuracy between P- and C-BlockTree.KOM just slightly differs for longer aggregation intervals, as opposed to the results for the mean relative node count error (cf. Figure 35a). Based on this observation, it is concluded that the increasing relative node count error of P-BlockTree.KOM does not only result from the decelerated collection and dissemination of data but from the loss of data at the different levels.

TRAFFIC AND POWER CONSUMPTION

The results for the mean upload and download traffic for both approaches are depicted in Figure 38 and outline the expected effect of the varied parameter on the overall upload and download traffic. A longer aggregation interval reduces the amount of periodically executed operations during a fixed time interval, leading to the depicted reduction in terms of traffic. Taking the results for five seconds, 10 s, 20 s, and 40 s as reference, the upload and download traffic halve. Finally, Figure 39



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 38: Overview on the overall traffic per node for a varying aggregation interval

depicts the cost in terms of the mean power consumption per node and represents the total effect of a varied aggregation interval on the arising cost. The mean power consumption summarizes and reflects the combined impact of the increasing or decreasing upload and download traffic.

Taking the previously discussed results for performance into consideration, it becomes apparent that an increasing accuracy and reduced staleness come at considerably increasing cost. Based on the results for the relative node count error, both approaches already provide accurate results for longer aggregation intervals, while preserving the scarce resources, such as bandwidth and energy. Moreover, the results reveal that the considerably increased traffic cannot be justified by an improved performance due to the small gain in terms of accuracy. For a reduction of the interval from 10 s to five seconds the traffic even doubles, while the accuracy of the provided monitoring results from P-BlockTree.KOM decreases. With the examination of the timeout factor and the parameter combination of the timeout and blocking interval factors, which are both presented in Appendix A.2.3 and A.2.5, other alternatives are introduced to adjust the performance at minimal cost.

6.3.2.5 *Summary*

After the evaluation of a fraction of selected system parameters, we review and summarize the obtained findings. The summary also takes the results from the variations of the remaining system parameters into consideration, which have not been discussed in this section but are presented in Appendix A.2. Based on the summary, we present the final configurations of P- and C-BlockTree.KOM for the following comparative evaluation with Mobi-G.KOM.

FINDINGS OF THE SYSTEM PARAMETER EVALUATION

In general, P- and C-BlockTree.KOM represent two decentralized monitoring mechanisms that successfully operate in MANETs and monitor the current state of the network, while handling moving as well as arriving or disappearing nodes. The default experiments consist of a scenario with and without churn, as introduced in Section 6.3.2.2. The corresponding results illustrate that both approaches exhibit a comparable performance in terms of accuracy and timeliness. They capture the current number of active nodes as well as the fluctuating UDSF attribute. P-BlockTree.KOM does not suffer from a larger hierarchy, because the staggered execution of

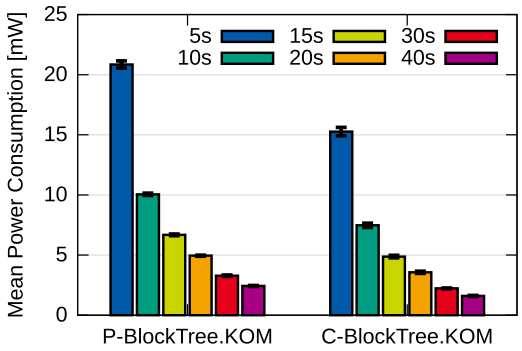


Figure 39: Mean power consumption for a varying aggregation interval

the Aggregating-Up operation during a single aggregation interval compensates the collection and dissemination over five instead of three levels. However, the comparable performance comes at the expense of a considerably higher traffic and power consumption. These preliminary findings indicate that C-BlockTree.KOM represents a more efficient approach than P-BlockTree.KOM due to the good performance at considerably lower cost.

Looking at the impact of the different parameters, we start with a summary of the variation of P- and C-BlockTree.KOM's monitoring topology, as presented in Section 6.3.2.3. The obtained results indicate that P-BlockTree.KOM benefits from a larger sector edge length, because the accuracy and the freshness of the monitoring results improve. Similarly, C-BlockTree.KOM profits from larger sectors, because the accuracy improves in terms of the node count. However, the results of the concentrating approach show that the positive impact of larger sectors mitigates for UDSF, whereas the staleness of results even increases. Taking the results for cost of both approaches into consideration, the observed gain in performance comes at an increased upload and download traffic. The applied communication methods of BlockTree.KOM do not compensate for the exchange of monitoring data between an increased number of sibling sectors. Particularly, they do not compensate for the concentration of monitoring data from blocks and/or sectors if the number of blocks and/or sectors per level increases.

In terms of the aggregation interval, as presented in Section 6.3.2.4, the results underpin the strong impact of this parameter on both approaches. The parameter has a direct influence on the trade-off between performance and cost. Shorter intervals improve accuracy and freshness at increased cost, whereas longer intervals reduce cost at a degrading performance. However, in terms of the node count, the results exhibit that the interval cannot freely be shortened. For very short intervals, the gain in accuracy mitigates, whereas the traffic still increases.

In contrast to the strong influence of the aggregation interval on the interdependence between performance and cost the timeout factor represents a cost-effective parameter. It influences the performance but does not affect the resulting traffic, as discussed in Appendix A.2.3. For smaller timeouts the accuracy considerably degrades, whereas longer timeouts improve performance. Based on the results for node count, the performance gain stems from a reduced underestimation due to the less aggressive deletion of stale entries from the hierarchy and result tables. In fact, long timeouts may be configured without risking the utilization of stale data, as stated in Appendix A.2.3. However, the configuration of longer timeouts is reflected by a

System parameter	P-BlockTree.KOM configuration	C-BlockTree.KOM configuration
Sector edge length	3	3
Aggregation interval	20 s	20 s
Leaf information timeout factor	1.2	1.2
Timeout factor	3.1	3.1
Blocking interval factor	0.9	1.2
Operation blocking interval factor	1.0	1.0

Table 27: Final configuration of P- and C-BlockTree.KOM's varied system parameters

degrading accuracy for UDSF and an increasing staleness and should be taken into consideration during the configuration of this parameter. Combined with the blocking interval factor, as presented in Appendix A.2.5, the simultaneous variation of both factors does not only improve accuracy, but even reduces the resulting cost, while accepting a slight degradation regarding UDSF and freshness. If the blocking interval factor is varied separately (cf. Section A.2.4), the increased traffic does not justify the small gain in performance for shorter blocking intervals. In contrast, longer intervals reduce the traffic but considerably degrade performance. If both parameters a varied simultaneously, (i) a longer blocking interval reduces the traffic and (ii) a larger timeout factor avoids the premature deletion of data from the tables due to the decelerated collection and dissemination.

SYSTEM PARAMETER CONFIGURATIONS OF P- AND C-BLOCKTREE.KOM

Based on the observed results and findings, P- and C-BlockTree.KOM are configured, as listed in Table 27. In terms of P-BlockTree.KOM, we configure the sector edge length with three sectors, because the gain in performance comes at minimal cost and communication takes place over a flat hierarchy. The results for the simultaneous variation of the blocking interval and timeout factor reveal that a comparable performance to the initial configuration is obtained at reduced cost. Though, we abstain from increasing the blocking interval factor but stick to the initial configuration of 0.9. We retain this configuration for an early transmission of messages and accept the higher cost in order to handle also dynamic scenarios. In terms of the timeout factor, 3.1 is chosen, because it improves the accuracy in terms of the node count, while the negative influence on UDSF and freshness is not that strong as for 6.1. Due to the previous decisions, which neglect the arising cost, we extend the aggregation interval to reduce the resulting upload and download traffic as well as the power consumption. For the leaf information timeout and operation blocking interval factor we stick to the default configuration, as the results of the corresponding parameter variations show that these configurations are already adequate (cf. Appendix A.2.2 and A.2.6).

Regarding C-BlockTree.KOM, we stick to the initial configuration of the sector edge length, because the outcome does not justify the configuration of longer sector edge lengths. The aggregation interval is prolonged and set to 20 s to reduce the burden on the resource-limited devices and to decrease the resulting traffic. Instead, we set the timeout factor to 3.1 to improve performance, without affecting the resulting traffic. Moreover, we even set the blocking interval factor to 1.2 instead of

0.9. The results for the combined parameter evaluation of the blocking interval and timeout factor uncover that the configuration of both factors does not only improve performance but even reduces cost in comparison to the default configuration. Due to this efficient combination we take the longer blocking interval for the dissemination of messages into account, which, however, might become a drawback in highly dynamic scenarios. Finally, we stick to the default configuration of the leaf information timeout and operation blocking interval factor. Similar to P-BlockTree.KOM they already represent reasonable configurations for both parameters.

6.3.3 *System Parameter Evaluation of* Mobi-G.KOM

Subsequent to the evaluation of C- and P-BlockTree.KOM, we complete the system parameter evaluation with the examination of Mobi-G.KOM's system parameters. Similar to the previous section we present the default configuration of Mobi-G.KOM for the experiments and identify relevant parameters as well as parameter combinations, using again a $2^k * r$ factorial experiment design, as described in [67]. Afterwards, we start with evaluating the impact of churn on performance and cost. The purpose of these experiments with and without churn is twofold. On the one hand, we show how Mobi-G.KOM operates in an optimal and normal scenario to use the obtained results as a reference for the subsequent system parameter evaluation. On the other hand, these experiments uncover the differences between Mobi-G.KOM with discrete and continuous monitoring and outline the consequences on performance and cost. To differentiate between the two variants we refer to Mobi-G.KOM with discrete monitoring as Mobi-G.KOM DM. The variant with continuous monitoring is denoted as Mobi-G.KOM CM.

6.3.3.1 *Default Configuration of* Mobi-G.KOM

Table 28 provides an overview about existing system parameters, how they are configured, and which parameters are varied during the evaluation. Dealing with the fixed parameters, the first four of them configure Mobi-G.KOM's communication methods. This includes the contention-based forwarding scheme for the dissemination of IS and army messages and the execution of response operations for missing or old information. Every node periodically refreshes its state about its current position every 10 s. The information about the position is used to determine the hesitation time, based on (7), with the corresponding hesitation factors that are described in (13) and (20), respectively. To increase the probability of a successful transmission of IC messages and to avoid lost or redundant tokens an IC message may be retransmitted up to three times. The corresponding acknowledgement timeout is set to 900 ms to reserve an adequate amount of time for the acknowledgement of a message before the next IC message is transmitted, which is configured by the token-send-delay parameter. The intermediate values threshold is set to 20 s to reduce the negative impact of new epochs on Mobi-G.KOM's accuracy and to offer a sufficient amount of time for the collection of new information at the beginning of an epoch.

The remaining three system parameters of Mobi-G.KOM, comprising epoch length, token-send-delay, and refresh timeout, represent the set of parameters that will be varied during the following evaluation. They have been chosen for the system parameter variation, because they are responsible for the configuration of data collection

Fixed system parameter	**Configuration**
Estimated maximum communication range	205 m
Maximum forwarding time	400 ms
Maximum responding time	1 s
Position refresh interval	10 s
Token acknowledgment timeout	900 ms
Number of IC message retransmissions	3
Intermediate values threshold	20 s
Varied system parameter	**Configuration**
Epoch length	5 min
Token-send-delay	1 s
Refresh timeout	1 s

Table 28: Overview on the default configuration of the fixed and varied system parameters of Mobi-G.KOM DM and CM

and result dissemination. Similar to BlockTree.KOM we start with an investigation of the effect of individual parameters as well as of parameter combinations, using a $2^k * r$ factorial experiment design. Contrary to the evaluation of BlockTree.KOM, we do not reduce the number of system parameters for the factorial experiment design but maintain the three identified parameters. Consequently, k is set to three and every experiment is repeated 10 times.

Table 29 shows how the three parameters are configured, using two configurations per parameter. For a better representation of the results we abbreviate epoch length as *EL*, token-send-delay as *TSD*, and refresh timeout as *RT*. Parameter combinations are indicated by the multiplication sign. The results for Mobi-G.KOM DM and CM are listed in Table 30 and 31. Similar to BlockTree.KOM the results reveal that both variants of Mobi-G.KOM are basically influenced by the variation of the individual parameters. The influence of parameter combinations is of minor importance and comes to a maximum of 10.39 % and 11.51 % for the combination of epoch length and token-send-delay for both variants. Due to the low relevance in contrast to the variation of the individual parameters we limit the system parameter evaluation to the three parameters, which is presented after the description of the impact of churn on Mobi-G.KOM.

6.3.3.2 *Influence of Churn on* Mobi-G.KOM

During this section, we investigate how Mobi-G.KOM behaves in an optimal scenario without churn and outline the influence of churn on the different components and aspects of the approach. The initial insights provide an overview on Mobi-G.KOM's performance and cost and serve as basis for the subsequent experiments. Moreover, the results highlight the differences between Mobi-G.KOM DM and CM.

Configuration	Epoch length (EL)	Token-send-delay (TSD)	Refresh timeout (RT)
Lower bound	3 min	1 s	1 s
Upper bound	7 min	16 s	16 s

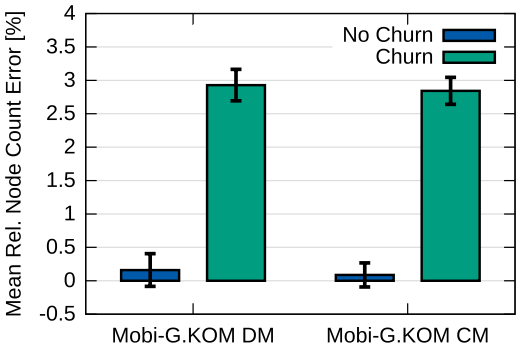
Table 29: Overview on the different configurations of the three chosen system parameters, which consist of two variations per parameter, representing the lower and upper bound for the configuration range.

System parameter	Proportion of variation [%]					
	Node count	UDSF	Staleness	Upload	Download	Power consumption
EL	6.87	98.11	98.73	1.76	0.52	0.66
TSD	60.72	0.13	0.00	0.45	0.00	0.01
RT	11.04	0.17	0.21	96.45	99.39	99.23
EL ∗ TSD	11.51	0.11	0.08	0.06	0.00	0.00
EL ∗ RT	1.01	0.50	0.22	0.10	0.01	0.01
TSD ∗ RT	5.36	0.05	0.00	0.23	0.02	0.04
EL ∗ TSD ∗ RT	0.56	0.00	0.00	0.05	0.00	0.04
Error	2.93	0.95	0.75	0.90	0.07	0.04

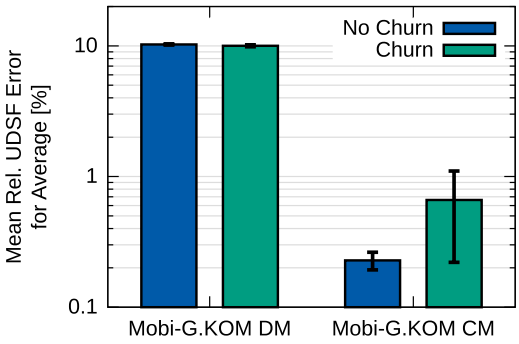
Table 30: Impact of different system parameters of Mobi-G.KOM DM on the identified metrics using a $2^k * r$ factorial experiment design

System parameter	Proportion of variation [%]					
	Node count	UDSF	Staleness	Upload	Download	Power consumption
EL	6.40	4.82	7.21	0.30	0.12	0.15
TSD	55.94	66.46	67.48	20.04	5.05	6.58
RT	14.16	20.84	18.38	76.56	90.94	89.73
EL ∗ TSD	10.39	0.21	0.81	0.02	0.00	0.00
EL ∗ RT	1.21	0.38	0.72	0.01	0.01	0.01
TSD ∗ RT	8.03	0.05	0.00	2.53	3.72	3.43
EL ∗ TSD ∗ RT	0.63	0.27	0.50	0.00	0.00	0.00
Error	3.25	6.97	4.90	0.53	0.14	0.08

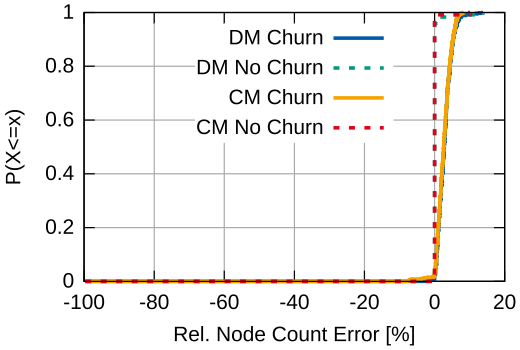
Table 31: Impact of different system parameters of Mobi-G.KOM CM on the identified metrics using a $2^k * r$ factorial experiment design



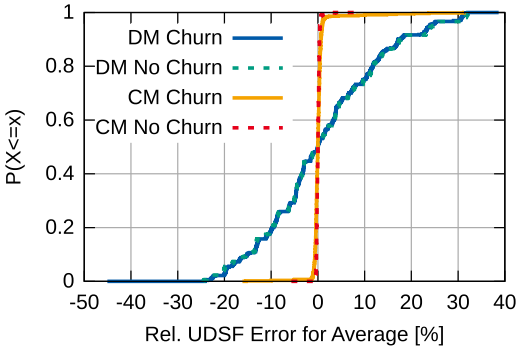
(a) Mean relative node count error



(b) Mean relative UDSF error using the average as aggregation function



(c) Cumulative distribution function of the relative node count error



(d) Cumulative distribution function of the relative UDSF error

Figure 40: Overview on accuracy of the provided monitoring results for node count and UDFS

ACCURACY

Starting with the examination of accuracy of the provided monitoring results, the plots in Figure 40 depict the impact of churn on node count and UDFS. In terms of the node count, Figure 40a shows the negative impact of arriving and leaving nodes on Mobi-G.KOM's accuracy. Whereas both variants of Mobi-G.KOM provide a nearly correct estimation of the active number of nodes with a mean error of 0.16 % and 0.09 %, respectively, both variants suffer from an increased error due to churn for two reasons. One problem arises from the fact that nodes with tokens leave the network so that these tokens are never reflected in the final global view of an attribute. In this case, Mobi-G.KOM underestimates the current state in the network. The second problem stems from the transmission of old tokens from nodes, which already left the network. Whereas tokens from new nodes are easily integrated into the global view of an attribute, Mobi-G.KOM can only remove old information from disappearing nodes by restarting an epoch. Until the end of an epoch the old information is always reflected in the global view of an attribute, which leads to an overestimation of the current state in the network. Taking a look at the distribution of the relative node count error (cf. Figure 40c)[3], the impact of churn becomes apparent. Whereas the impact of leaving nodes with tokens becomes not yet visible due to the small interval for the transmission of IC messages, both variants of Mobi-G.KOM

3 In Figure 40c and 40d, Mobi-G.KOM DM is abbreviated as *DM* and Mobi-G.KOM CM as *CM*.

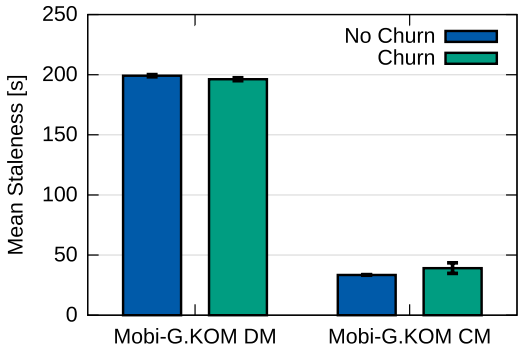


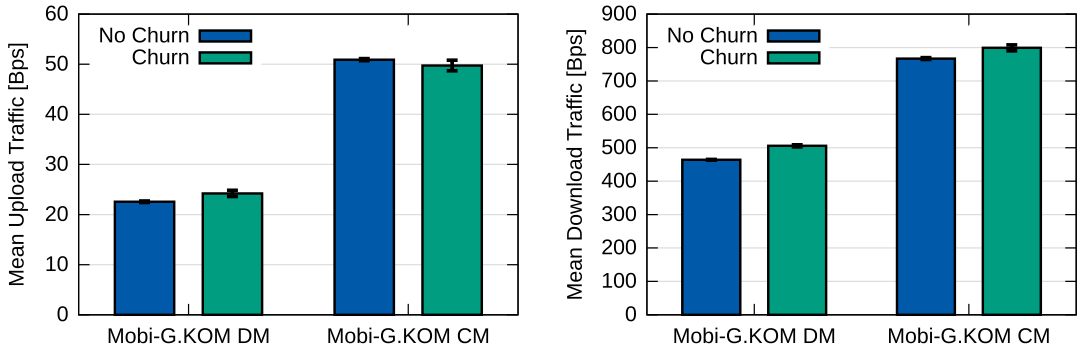
Figure 41: Mean staleness of the provided monitoring results

overestimate the current state. The overestimation accounts for the increased mean relative node count error and arises from the collection of data from disappeared nodes. Considering the accuracy for monitoring UDSF, the relative error remains nearly constant for Mobi-G.KOM DM, while it slightly increases for Mobi-G.KOM CM. In general the reason for the small impact of churn on monitoring the UDFS, results from the fact that a missing or new value of the attribute hardly influences the monitored global view, especially when the average is applied as aggregation function.

Apart from the effect of churn, Figure 40 outlines the differences between the two variants of Mobi-G.KOM in terms of accuracy. Whereas Figure 40a reveals that there is no significant difference between both variants in terms of the mean relative node count error, Figure 40b highlights the advantage of Mobi-G.KOM CM on accuracy if fluctuating attributes are monitored. In terms of discrete monitoring, Mobi-G.KOM uses the values from the beginning of an epoch as input to calculate the current global view of an attribute. During the whole epoch, it operates on these values and disseminates the resulting global view to all nodes, whereas the effective global view of the attribute changes. As a result the mean error accounts for 10.25 % and 10.02 % without and with churn, respectively. In contrast, continuous monitoring enables the integration of new values during the whole epoch. As a consequence, the global view is constantly updated, leading to the reduced mean relative error of 0.23 % and 0.66 % without and with churn, respectively. Figure 40d depicts the distribution of the relative error for the UDSF attribute and reveals that Mobi-G.KOM DM does not correctly capture the attribute and that the majority of obtained monitoring results deviates from the effective global state. Contrary to these results, Mobi-G.KOM CM provides almost entirely correct results where 95 % of the monitoring results exhibit a relative error smaller than 1.09 % with churn and 0.46 % without churn, when considering the absolute value of the relative error.

STALENESS

Figure 41 depicts the mean staleness of the monitoring results, which outlines that churn has little impact on the freshness of the results. Regarding Mobi-G.KOM DM, the mean staleness decreases from 199.16 s to 196.23 s, whereas it increases for Mobi-G.KOM CM from 33.49 s to 39.14 s. The decrease of the mean staleness in terms of Mobi-G.KOM DM arises from the integration of new tokens during an epoch. Without churn the tokens are created during the restart of an epoch, whereas with churn, nodes arrive during an epoch and integrate their new tokens, which



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 42: Overview on the overall mean traffic per node

slightly reduces the overall staleness. The reason for the increase of the mean staleness in terms of Mobi-G.KOM CM arises from the unsuccessful forwarding of tokens to the collecting beacon, because nodes along the path to the beacon may leave the network. If the leaving node currently holds a single or multiple tokens with updates from monitored attributes, the corresponding information is lost, which leads to an increasing error and staleness.

Focusing on the differences between the two variants of Mobi-G.KOM regarding staleness, we observe that the outcome for staleness exhibits similar tendencies as the outcome for the mean relative UDSF error. Since Mobi-G.KOM DM uses the measured values of attributes from the beginning of an epoch, the staleness of information increases during the epoch, because no updates are provided. Using continuous monitoring, Mobi-G.KOM periodically updates the values and spreads the updated monitoring results in the network, which leads to the reduced staleness.

TRAFFIC AND POWER CONSUMPTION

Dealing with cost, Figure 42 depicts the corresponding results for the overall upload and download traffic. The plots reveal that the high accuracy and freshness of continuous monitoring come at the expense of an increased upload and download traffic. The reason for this increase stems on the one hand from the continuous creation and transmission of tokens. In contrast, the continuous integration leads to a continuous result dissemination, which accounts for more than half of the total traffic. Contrary to this behavior, Mobi-G.KOM DM transmits the majority of tokens at the beginning of a new epoch. As a result the amount of IS messages is reduced after having disseminated the aggregated monitoring results for the first time, as indicated by the reduced total traffic. A detailed examination of the total upload and download traffic and the classification into the different traffic types is found in Appendix A.3, which underpins the statements about the arising traffic of IC as well as IS messages.

Regarding churn, the results slightly reveal the interesting effect of an increased node density paired with the contention-based forwarding scheme on the upload and download traffic, as already observed and discussed during the system parameter evaluation of BlockTree.KOM (cf. Section 6.3.2.2). Especially when evaluating the upload and download traffic of Mobi-G.KOM CM, which is dominated by the IS traffic that relies on the presented contention-based forwarding scheme, it becomes apparent that the upload traffic decreases, whereas the download traffic increases.

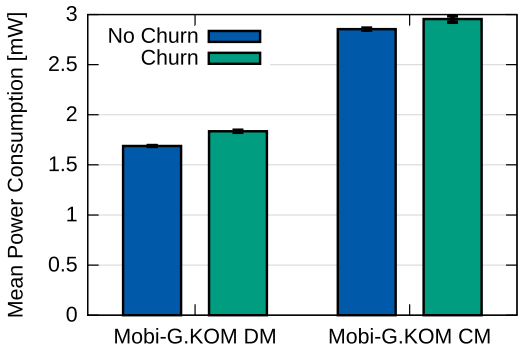


Figure 43: Mean power consumption per node

As already detailed for BlockTree.KOM the current number of active nodes is most of the time above the configured number of nodes in the scenarios with churn, which leads to a slightly increased node density. The higher density reduces the effective mean upload traffic per node, whereas the download traffic increases due to the reception of new or redundant IS messages. This effect becomes more obvious and even observable for Mobi-G.KOM DM during the comparative evaluation, where the node density is varied (cf. Section 6.4.1.2).

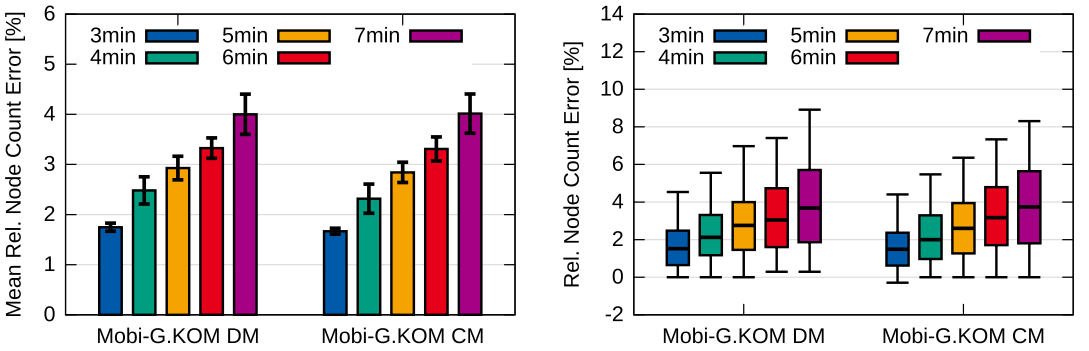
The observed tendencies for the overall traffic become also apparent by the results for the mean power consumption, which is depicted in Figure 43. Similar to the evaluation of BlockTree.KOM the results do not comprise the overall mean power consumption but only the supplementary consumption, which arises from the transmission and reception of data. The good performance of continuous monitoring, especially when monitoring fluctuating attributes, comes at the expense of an increased mean power consumption. In this context, the results for churn reveal that the reduced amount of sent messages from Mobi-G.KOM CM does not suffice to compensate the increased traffic so that the mean power consumption increases.

6.3.3.3 *Epoch Length*

After examining the influence of churn on the default system configuration, this section starts with the evaluation of Mobi-G.KOM's relevant system parameters and contains the results for varying the epoch length. The remaining system parameters are configured, as listed in Table 28, whereas the different configurations of the epoch length are shown in Table 32. Based on the default configuration, which is highlighted by the underlined value, the duration is decreased and increased to examine the impact of shorter and longer epochs on the performance and traffic. The two configurations of this parameter, which have been initially specified during the $2^k * r$ factorial experiment design, serve as lower and upper bound for the configuration range in these experiments. As outlined by the following results the linear

System parameter	Configuration
Epoch length [min]	3, 4, 5, 6, 7

Table 32: System parameter variation of epoch length. The underlined value represents the default configuration of the system parameter.



(a) Mean relative node count error

(b) Distribution of the relative node count error of all collected measurements

Figure 44: Overview on the mean relative node count error of the provided monitoring results for a varying epoch length

decrease and increase of the parameter already suffice to examine the impact on both variants of Mobi-G.KOM.

ACCURACY AND STALENESS

Figure 44 shows the influence of the epoch length on the mean relative node count error. As already outlined in the previous section and supported by the presented results both variants of Mobi-G.KOM attain a comparable performance in terms of this attribute. Both variants are not able to counteract (i) the loss of tokens due to disappearing nodes as well as (ii) the collection of transmitted tokens from disappearing nodes. An adjustment of the monitored state is only possible by restarting the epoch so that each node starts with a new token and, most important, that old tokens are removed from the results to prevent an overestimation of the active number of nodes. Consequently, the results in Figure 44a reveal that a shorter epoch with earlier restarts significantly reduces the monitoring error of both variants, whereas longer epochs increase this error. As discussed in the previous section the increasing error basically arises from an overestimation due to the collection of tokens from disappeared nodes. Figure 44b shows the distribution of the relative monitoring error per experiment and underlines this observation. It becomes apparent that the accuracy of both variants decreases, because the majority of results overestimates the current situation in the network. If epochs are shorter, Mobi-G.KOM reacts faster on stale tokens and avoids that the number of stale tokens accumulates over time.

Similar tendencies are observable when examining the impact of the epoch length on monitoring UDSF, as shown in Figure 45. However, the influence on both variants of Mobi-G.KOM varies. For Mobi-G.KOM DM the accuracy significantly increases for a shorter epoch length, whereas a longer epoch length amplifies the effect of operating on values from the beginning of an epoch. Dealing with Mobi-G.KOM CM, the results only indicate a tendency towards an improved accuracy but without a statistical significance. In fact, continuous monitoring enables to capture fluctuating attributes even for a longer epoch length and counteracts the loss or transmission of stale tokens so that the negative influence of longer epochs is mitigated. As depicted in Figure 46 the results for staleness reflect the observed tendencies for UDSF for both variants of Mobi-G.KOM. The utilization of attributes from the beginning of

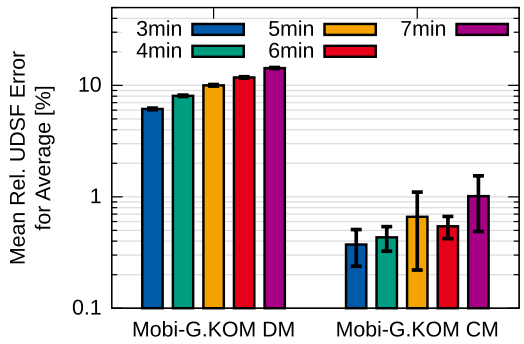


Figure 45: Mean relative UDSF error using the average as aggregation for a varying epoch length

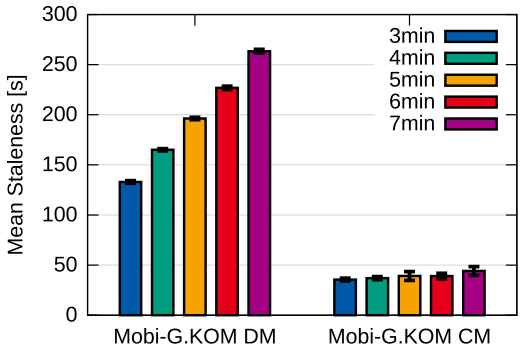


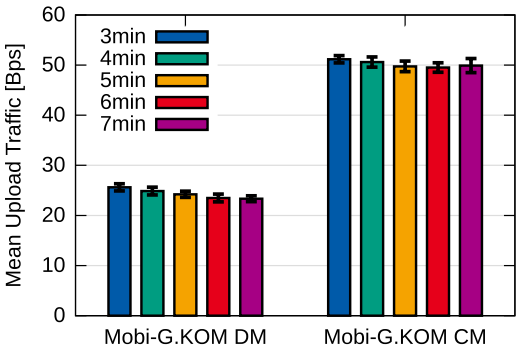
Figure 46: Mean staleness of the provided monitoring results for a varying epoch length

an epoch leads to a considerably increasing staleness as a function of the epoch length. In contrast, the constant collection of new values counteracts longer epochs and reduces the negative influence of them on the freshness of results.

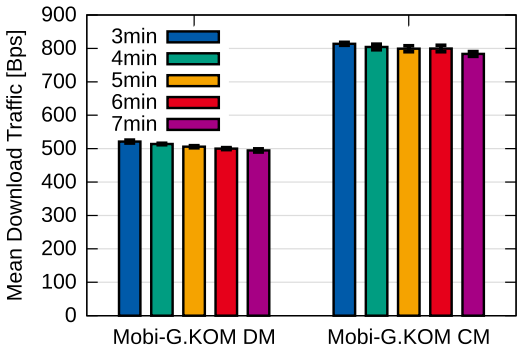
TRAFFIC

The results for the mean traffic reveal the interesting insight that the positive impact of a decreasing epoch length on accuracy and staleness does not come at the expense of increased cost (cf. Figure 47). Instead, the outcome indicates that shorter epochs represent a cost-effective system parameter to improve accuracy and freshness. In the following we describe the reasons for the small impact of the epoch length by separately examining the influence on the different traffic types. The corresponding results are depicted in Figure 48 and show the differences between the varied and default configuration of epoch length. This representation highlights the small impact of the system parameter and additionally outlines if two configurations are significantly different. In this case the confidence intervals do not intersect with the x-axis.

For both variants of Mobi-G.KOM Figure 48c shows the small impact of the parameter on the resulting traffic for the identification of the strongest army and its corresponding beacon. Though, shorter epoch lengths lead to a higher frequency of identifying the beacon and the strongest army, the traffic is only slightly influenced,

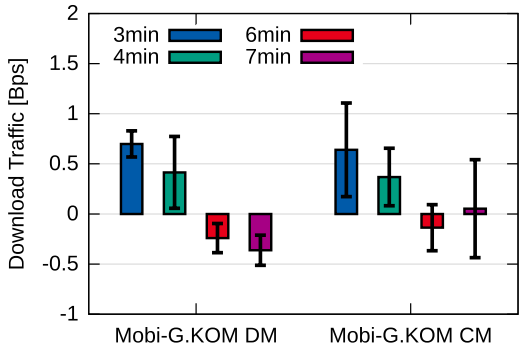


(a) Mean overall upload traffic per node

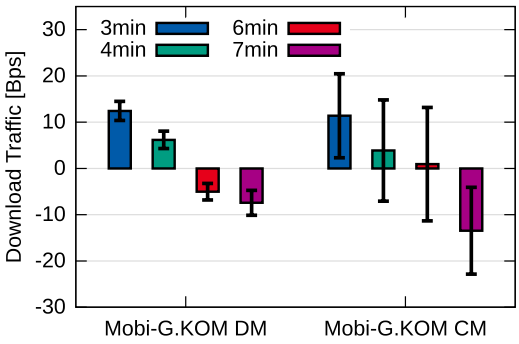


(b) Mean overall download traffic per node

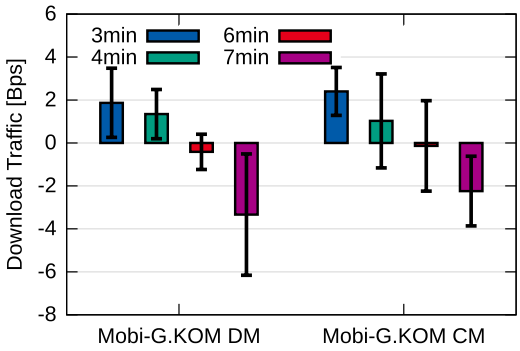
Figure 47: Overview on the overall traffic per node for a varying epoch length



(a) Difference between the varied and default configuration of the epoch length parameter regarding the mean download traffic per node for the transmission of IC messages



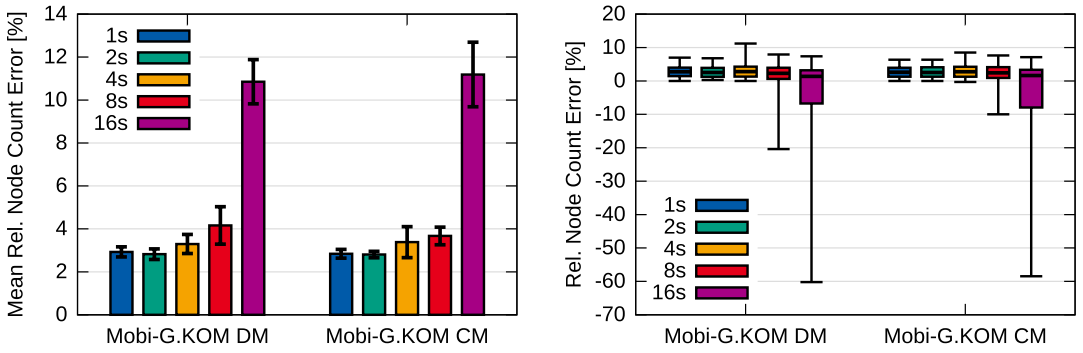
(b) Difference between the varied and default configuration of the epoch length parameter regarding the mean download traffic per node for the transmission of IS messages



(c) Difference between the varied and default configuration of the epoch length parameter regarding the mean download traffic per node for the transmission of army messages

Figure 48: Overview on the differences between the varied and default configuration of the epoch length parameter regarding the download traffic per node for the transmission of IC, IS, and army messages

which leads to the conclusion that this identification just accounts for a small fraction of the total army traffic. The larger fraction of the traffic is attributed to the periodic dissemination of army message to update, for instance, the shortest path towards the beacon. In terms of Mobi-G.KOM DM, Figure 48a reveals that a shorter or longer epoch length leads to a significantly increased or reduced IC traffic, respectively. However, its influence on the overall traffic is negligible due to the small amount of IC traffic, especially, when taking the arising IS and army traffic into consideration. The IS traffic also varies as a function of the epoch length (cf. Figure 48b). Nevertheless, as the dissemination of the current state is mainly restricted to the beginning of an epoch, the impact on the download traffic per node is significant but small. In terms of Mobi-G.KOM CM, the influence of the epoch length on the IC and IS traffic is even smaller and only significant for two of the four variations, since IC and IS messages are constantly transmitted during the whole epoch.



(a) Mean relative node count error

(b) Distribution of the relative node count error of all collected measurements

Figure 49: Overview on the mean relative node count error of the provided monitoring results for a varying token-send-delay

6.3.3.4 *Token-Send-Delay*

In this section we study the influence of different configurations of the token-send-delay parameter on Mobi-G.KOM. Table 33 lists the different configurations for the system parameter variation. The underlined value represents the default configuration, which is doubled during the different experiments until the upper bound of 16 s is reached. The variation just comprises a growing token-send-delay, because the default configuration already constitutes a lower bound of the parameter's configuration range, as confirmed by the following evaluation. In contrast to the linear variation of the epoch length we double the values to highlight and emphasize the influence of the parameter on both variants of Mobi-G.KOM.

System parameter	Configuration
Token-send-delay [s]	1, 2, 4, 8, 16

Table 33: System parameter variation of the token-send-delay. The underlined value represents the default configuration of the system parameter.

ACCURACY AND STALENESS

The influence of an increasing token-send-delay parameter on the mean relative node count error is depicted in Figure 49. The results in Figure 49a reveal that a parameter configuration below four seconds has no significant impact on the error, which does not decrease but remains constant. A configuration of two seconds suffices as well for the timely collection of monitoring data to provide recent and accurate results for our periodic measurements, which we take every minute from all active nodes. Above a configuration of two seconds, the increasing error is explained by looking at the impact of a longer token-send-delay on the token collection. Figure 50 depicts the mean duration of a token to get from the creating node to the collecting beacon and reveals that a larger token-send-delay results in a longer token collection time. This negative influence becomes particularly apparent when considering the mean relative node count error (cf. Figure 49a), where the error increases for larger token-send-delays. The reason for the collapse results from the fact that a longer sojourn

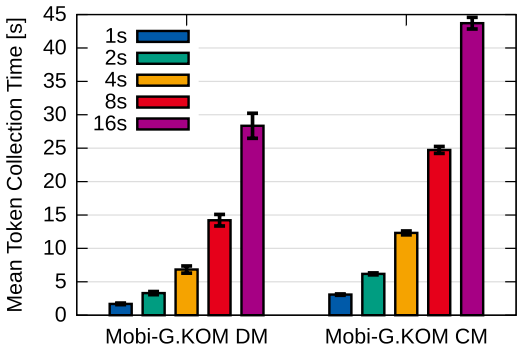
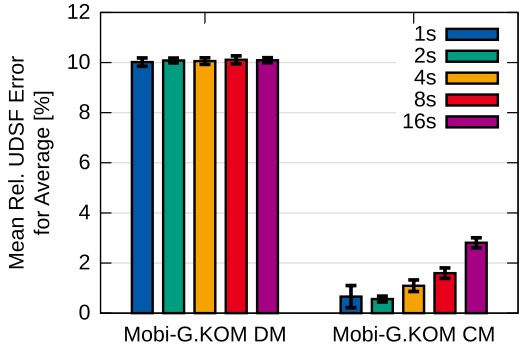


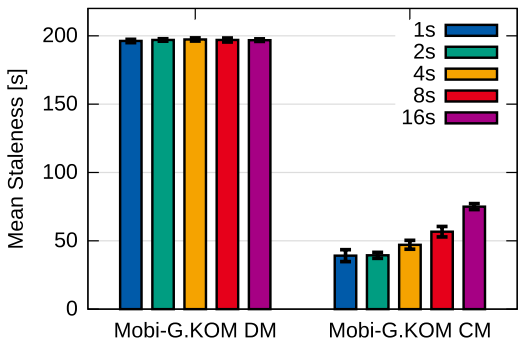
Figure 50: Mean duration for a token to get from the creating node to the collecting beacon

time of tokens at forwarding nodes increases the probability that these tokens are lost due to disappearing nodes. As shown by Figure 49b both variants of MOBI-G.KOM underestimate the current state of the network, which becomes particularly apparent for a token-send-delay of eight second and 16 s.

The results for the discrete monitoring of UDSF constitute an exception (cf. Figure 51a), because the mean relative monitoring error is hardly influenced by the system parameter. This results from the fact that lost values of the attribute hardly influence the monitored global view. Especially when applying the average as aggregation function, the influence of incomplete monitoring data on the monitoring results mitigates. MOBI-G.KOM DM relies on the locally measured values of attributes from the beginning of an epoch. Consequently, a larger token-send-delay just delays the collection of tokens at the beginning of an epoch and exhibits no influence on the relative UDSF error. In contrast, a negative impact on MOBI-G.KOM CM is observable, because a larger token-send-delay decelerates the constant collection of local values, which leads to an increased deviation of the monitored state from the effective one. The outcome for the mean staleness supports this observation (cf. Figure 51b), because an increasing token-send-delay does not change the high staleness of monitoring results for discrete monitoring. In contrast, the impact of the decelerated collection on continuous monitoring is reflected by the increasing staleness.

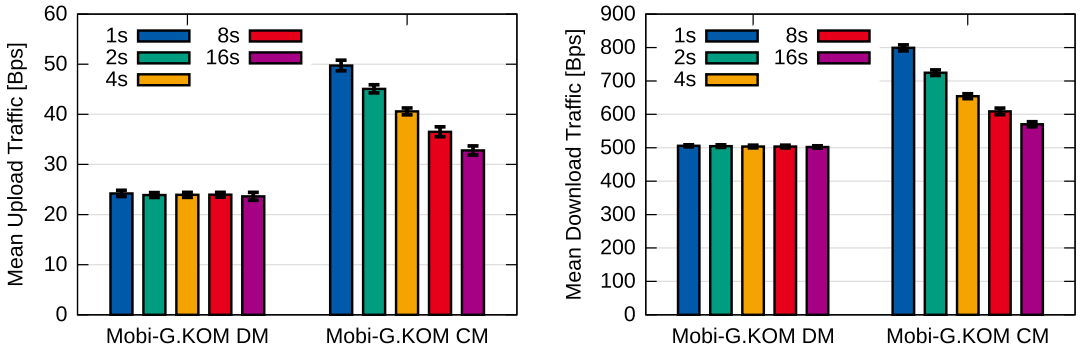


(a) Mean relative UDSF error using the average as aggregation function



(b) Mean staleness of the provided monitoring results

Figure 51: Overview on the mean relative UDSF error and the mean staleness of the provided monitoring results for a varying token-send-delay



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 52: Overview on the overall traffic per node for a varying token-send-delay

TRAFFIC

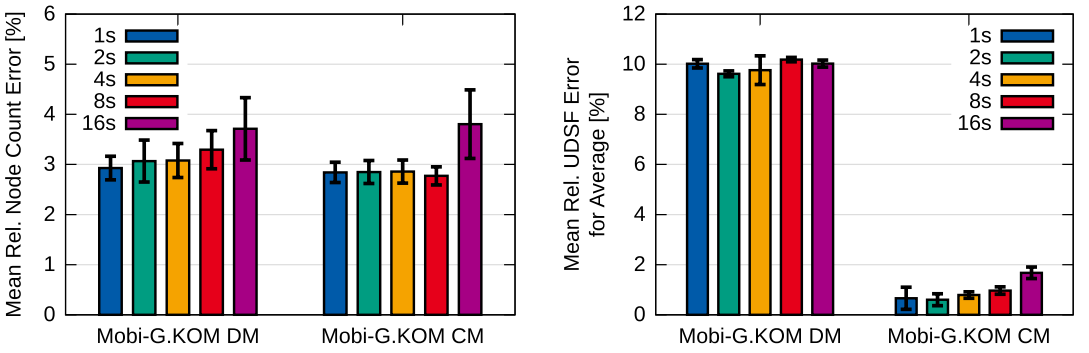
The obtained results for traffic in Figure 52 reveal that the impact of an increasing token-send-delay for both variants of Mobi-G.KOM resembles the outcome for staleness and the mean relative UDSF error, as discussed before. A variation of the system parameter hardly influences the upload and download traffic for discrete monitoring, whereas an increasing token-send-delay decreases the arising upload and download traffic of Mobi-G.KOM CM (cf. Figure 52a and 52b). A larger token-send-delay reduces the number of token-cycles per epoch, thus, the slots for the transmission of tokens are reduced as well. Mobi-G.KOM DM just relies on a fraction of cycles at the beginning of an epoch to collect the tokens, whereas the majority of cycles remains unused and does not influence the upload and download traffic. Contrary to this observation, the reduced number of cycles per epoch reduces the transmission and collection of the continuously created tokens by Mobi-G.KOM CM. Furthermore, the lower update frequency also reduces the frequency to disseminate the updated monitoring results in the MANET, which becomes apparent by the overall traffic reduction. The results for the mean power consumption are omitted, because they resemble the tendencies of the overall mean upload and download traffic.

6.3.3.5 *Refresh Timeout*

To complete the system parameter evaluation of Mobi-G.KOM this section deals with the variation of refresh timeout, which defines the length of the information-cycle that triggers the periodic transmission of IS and army messages. Table 34 lists the corresponding configurations, which are the same as for the token-send-delay. One second represents the default parameter configuration and constitutes the lower bound of the configuration range. During the parameter variation, the value is doubled up to a maximum of 16 s to highlight the influence of the system parameter on Mobi-G.KOM.

System parameter	Configuration
Refresh timeout [s]	1, 2, 4, 8, 16

Table 34: System parameter variation of the refresh timeout. The underlined value represents the default configuration of the system parameter.



(a) Mean relative node count error

(b) Mean relative UDSF error using the average as aggregation function

Figure 53: Overview on accuracy of the provided monitoring results for a varying refresh timeout

ACCURACY

Figure 53 depicts the outcome of the mean relative error and reveals the small impact of the refresh timeout on accuracy of both variants of MOBI-G.KOM. With respect to the mean relative node count error (cf. Figure 53a) only a configuration of 16 s of refresh timeout exhibits an observable impact, as the mean relative error increases. Similar to the results for the token-send-delay this impact mitigates for the discrete monitoring of UDSF (cf. Figure 53b), due to (i) the application of the average as aggregation function as well as (ii) the utilization of locally measured values from the beginning of an epoch.

To explain the overall small influence on accuracy of MOBI-G.KOM we take a closer look at the procedures that are influenced by the system parameter. The procedures comprise the periodic transmission of IS and army messages, which are both triggered by the information-cycle that is configured by the refresh timeout. Figure 54 depicts the influence of the parameter on the identification of a beacon and shows the mean duration from the advent of the strongest or impervious beacon until its conquest of the whole MANET. Figure 55 shows the mean duration for a monitoring result to get from a beacon to a node as a function of the refresh timeout. For the first case it becomes apparent that the duration for the conquest of the whole network considerably increases, which also influences the creation and collection of tokens, because a node always creates a new token if it joins a new army. Nevertheless, the extended time to conquest the network has little impact on accuracy, because the aggressive configuration of the token-send-delay ensures a fast and timely collection of monitoring data (cf. Figure 50) that counteracts the prolonged conquest time. Dealing with the mean information dissemination time in Figure 55, the impact of the varied parameter on the dissemination time is not that strong as on the time to identify the beacon or, in terms of the token-send-delay, on the time to collect the tokens. As a result the accuracy of the monitored results just slightly degrades, due to the small impact of the parameter on the time to disseminate the global view of aggregates.

Similar to a longer token-send-delay a longer refresh timeout increases the probability that a node with recent monitoring results leaves the network. However, this circumstance does not state a problem, because MOBI-G.KOM exploits the fact that

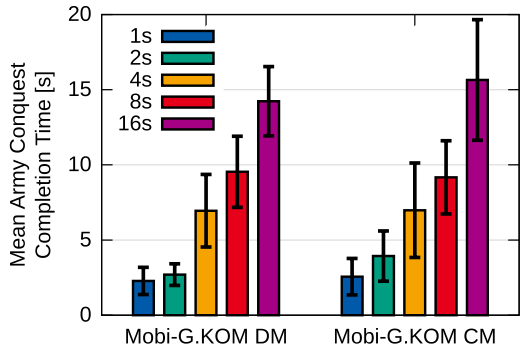


Figure 54: Mean time to conquer the whole network

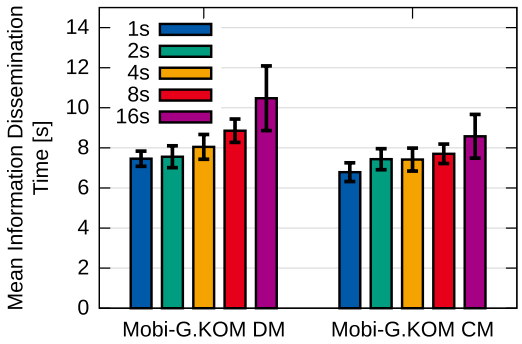


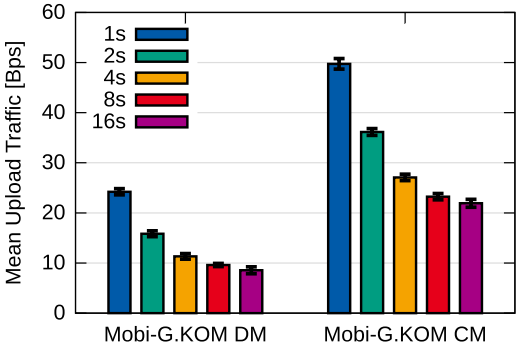
Figure 55: Mean information dissemination time

there are multiple candidates to forward IS messages. Whereas tokens disappear with their leaving nodes, other candidates step in to forward IS messages. The results for the impact of staleness are omitted, because they exhibit the same characteristics as the results for the mean relative monitoring error.

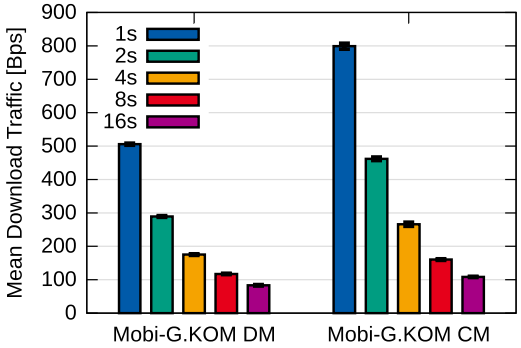
TRAFFIC

Whereas the refresh timeout has only small influence on Mobi-G.KOM's performance in terms of accuracy and staleness, it exerts a considerable and significant influence on the arising traffic, as shown in Figure 56. Both variants benefit from a longer refresh timeout, because the upload and download traffic decreases, whereas Mobi-G.KOM's accuracy and staleness mostly remain unaffected.

To investigate the cause for the immense impact of the parameter on the overall traffic we take a closer look at the different traffic types. Figure 57 depicts the download traffic for the exchange of IS and army messages (cf. Figure 57a and 57b), which constitute the two traffic types that are influenced by refresh timeout. Similar to the token-send-delay the refresh timeout influences the number of cycles and sending slots for both types of messages. An increasing refresh timeout leads to a reduced amount of cycles so that fewer messages are transmitted. Since both IS and army messages are transmitted throughout the whole epoch, a reduction of the sending slots has a considerable impact on the traffic, which significantly decreases with an increasing refresh timeout. Figure 57b additionally highlights the fact that the arising

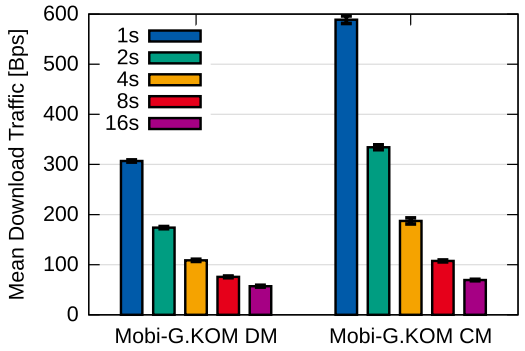


(a) Mean overall upload traffic per node

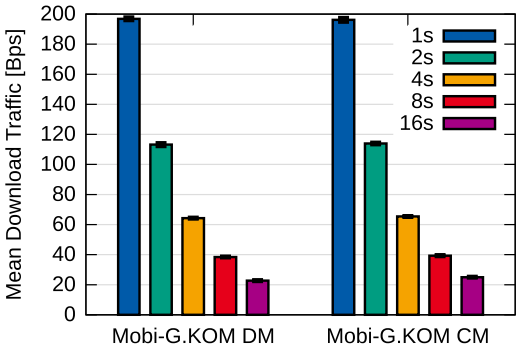


(b) Mean overall download traffic per node

Figure 56: Overview on the overall traffic per node for a varying refresh timeout



(a) Mean download traffic per node for the transmission of IS messages



(b) Mean download traffic per node for the transmission of army messages

Figure 57: Overview on the IS and army download traffic per node for a varying refresh timeout

army traffic is not influenced by discrete or continuous monitoring. As discussed in Appendix A.3 the identification of the strongest army with its corresponding beacon does not depend on the type of monitoring with the related information exchange. The results for the mean power consumption are omitted, because they resemble the tendencies of the overall mean upload and download traffic.

6.3.3.6 *Summary*

For the completion of Mobi-G.KOM's system parameter evaluation, we reflect and summarize the results and findings from the executed experiments. Afterwards, we detail the final configurations of Mobi-G.KOM DM and CM for the following comparative evaluation with BlockTree.KOM.

FINDINGS OF THE SYSTEM PARAMETER EVALUATION

In general, Mobi-G.KOM represents an accurate decentralized monitoring mechanism and offers two variants with different characteristics to monitor a MANET. Dependent on the demands of an application scenario, Mobi-G.KOM CM captures even fluctuating attributes and provides fresh and accurate results. The observed performance comes at increased cost, which are considerably reduced by relying on Mobi-G.KOM DM if (i) limited resources of a mobile device must be preserved or (ii) highly accurate results of fluctuating attributes are not required. Based on the results for monitoring the node count, it becomes apparent that the two variants of Mobi-G.KOM do not differ and provide comparable results.

The results for the different configurations of the epoch length in Section 6.3.3.3 reveal that this system parameter represents a cost-effective alternative to improve accuracy and freshness at minimal cost. Especially, Mobi-G.KOM DM benefits from this variant as the error for capturing highly fluctuating attributes is reduced with a small impact on the resulting traffic and power consumption. In terms of monitoring the current number of nodes, the accuracy increases for both variants, because an earlier restart of the epoch prevents both variants from overestimating the current state.

Contrary to the epoch length, which improves the performance at nearly constant cost, refresh timeout represents a parameter that preserves the resources of

a mobile node and reduces the arising traffic without affecting the performance of Mobi-G.KOM. The results in Section 6.3.3.5 reveal that the longer interval for the transmission of IS and army messages suffices to provide accurate and fresh results. Paired with an aggressive configuration of the token-send-delay the effect of an extended duration for the identification of a beacon is counteracted. Moreover, the influence of the parameter on the result dissemination time is small, because the utilized contention-based forwarding scheme enables the immediate and robust forwarding of results. Furthermore, a configuration with a longer refresh timeout reduces the slots for the transmission of IS and army messages so that the traffic and the resulting impact on the power consumption are minimized.

Finally, the token-send-delay constitutes a parameter that influences both the performance and cost of Mobi-G.KOM, as outlined in Section 6.3.3.4. The results reveal that a larger token-send-delay reduces the cost at the expense of a decreasing accuracy depending on the considered variant of Mobi-G.KOM as well as of the monitored attribute. In contrast, configurations with smaller values increase the traffic but do not necessarily result in a higher accuracy. As shown by the results for the token-send-delay with a configuration of one and two seconds they do not influence the periodic measurements for our evaluation. We take these measurements with an interval of one minute to assess the performance of our monitoring mechanisms. If we vary the interval for the measurements, the influence of the token-send-delay changes. As a result the parameter should be configured according to the planned or estimated access interval of the results.

SYSTEM PARAMETER CONFIGURATIONS OF MOBI-G.KOM DM AND CM

Based on the outcome, the considered system parameters are configured as shown in Table 35, which resembles the configuration in our previous work [160]. With an epoch length of three minutes we exploit the good impact on accuracy and freshness at minimal cost. The token-send-delay is set to two seconds to reduce the duration of a token inside the network as well as to minimize the probability that tokens get lost due to disappearing nodes. Finally, the configuration of the refresh timeout deviates from our previous work, because we set the parameter to four instead of eight seconds, sacrificing the resources of the mobile nodes to obtain accurate and fresh results even in more dynamic scenarios.

System parameter	**Configuration**
Epoch length	3 min
Token-send-delay	2 s
Refresh timeout	4 s

Table 35: Final configuration of Mobi-G.KOM's varied system parameters

6.4 COMPARATIVE EVALUATION

Subsequent to the system parameter evaluation, we conduct a comparative evaluation of the developed monitoring mechanisms. During the evaluation, we examine how our approaches behave in experiments with different scenario settings and var-

ied scenario parameters. The design of the scenario parameter variations for the comparative evaluation is based on the workload-dependent non-functional requirements, which address scalability and robustness of a decentralized monitoring mechanism (cf. Section 2.4.3). In terms of scalability, we examine how the monitoring mechanisms perform in MANETs of different spatial sizes and handle sparsely or densely populated areas. Subsequently, we assess robustness and vary different scenario parameters to model the characteristics of MANETs. During these experiments, we investigate how the monitoring mechanisms perform while handling the dynamic nature of MANETs. Finally, we evaluate how the mechanisms handle the distance-aware relevance of information and provide location-aware monitoring results.

6.4.1 *Evaluating Scalability*

The evaluation of scalability is divided into two parts: the first part investigates the influence of the spatial size of a network on performance and cost, whereas the second part examines the impact of node density. In the following section we focus on the spatial network size. Subsequently, Section 6.4.1.2 analogously examines node density.

6.4.1.1 *Variation of the Spatial Network Size*

For an isolated investigation of the influence of the spatial network size the edge length of the corresponding square area as well as the number of nodes inside the area must be varied. Both scenario parameters are altered to obtain the same node density in every experiment and to study only the influence of the spatial size. Table 36 lists the respective configurations for the two parameters. In addition, the table shows the maximum hierarchy level l_{max} of C- and P-BlockTree.KOM to cover the whole MANET. Both the modeled square areas and l_{max} of C- and P-BlockTree.KOM are depicted in Figure 58. The figure shows the logical partitioning of the area into blocks and the resulting sectors at different levels[4]. Due to the fact that both approaches are configured with a sector edge length of three the resulting sectors are situated at the same position and congruent. However, the concentrating blocks from C-BlockTree.KOM are highlighted by colored blocks to indicate where data are concentrated at the respective levels.

Scenario parameter	**Configuration**					
Edge length [m]	145	435	1000	1305	2000	3915
Number of nodes	4	33	175	297	698	2673
System parameter	**Configuration**					
l_{max} of P- and C-BlockTree.KOM	0	1	2	2	3	3

Table 36: Parameter variation of the spatial network size with the specification of the resulting maximum hierarchy level l_{max} of both approaches

Taking the default scenario for the system parameter evaluation as basis, we scale the spatial size in both directions. Due to the sensitivity of C-BlockTree.KOM re-

4 The sectors at l_1 are omitted to simplify the figure.

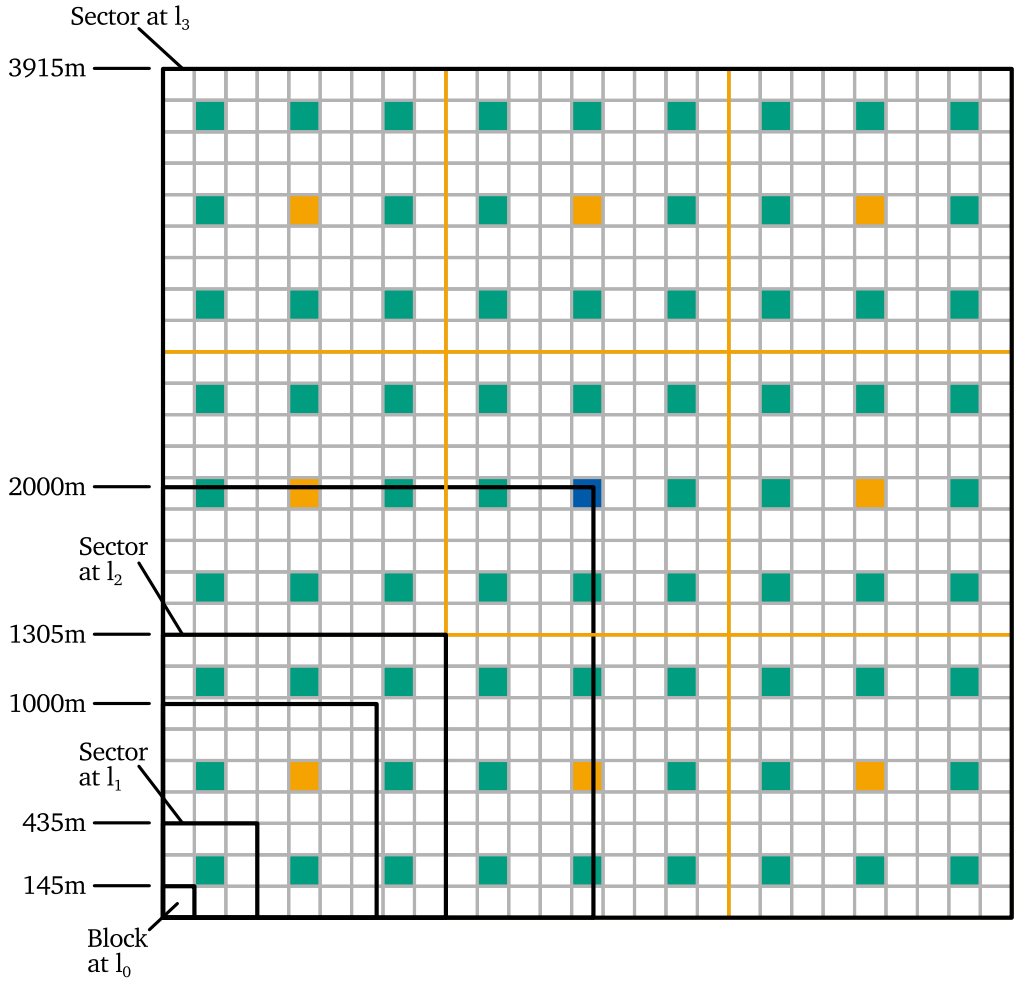


Figure 58: Planar visualization of the monitoring topologies of P- and C-BlockTree.KOM and of the modeled area for different configurations of the edge length. The colored blocks highlight the concentrating blocks of C-BlockTree.KOM.

garding the spatial network size we set the edge length in such a way that the underlying hierarchy of C-BlockTree.KOM is properly established. With a length of 145 m, 435 m, 1305 m, and 3915 m C-BlockTree.KOM establishes a complete hierarchy, where the highest sector exactly covers the modeled area, as depicted in Figure 58. The additional configurations of 1000 m and 2000 m are randomly set to outline the effect of an independently chosen square area on the approaches and particularly on C-BlockTree.KOM.

ACCURACY AND STALENESS

Starting with the results for P-BlockTree.KOM, Figure 59 shows that the mean relative node count error decreases as a function of the spatial network size, when ignoring the results for the smallest experiment that just consists of one block. The reason for the decreasing mean relative error is a reduced overestimation for larger spatial networks. The overestimation is primarily attributed to the prolonged timeout for the deletion of data from the hierarchy and result tables. The longer timeout is beneficial for scenarios with a larger spatial area, where monitoring data are col-

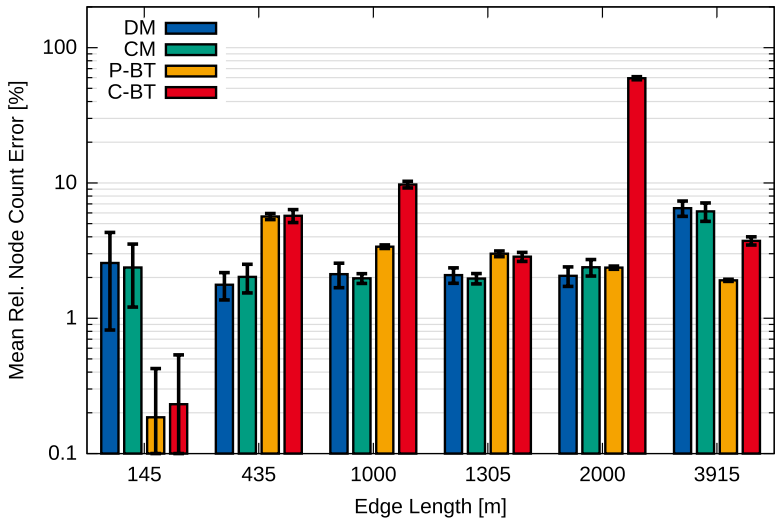
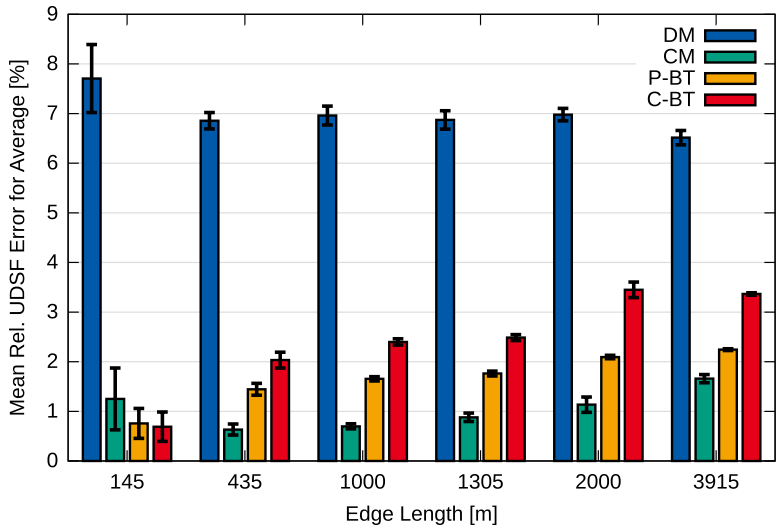


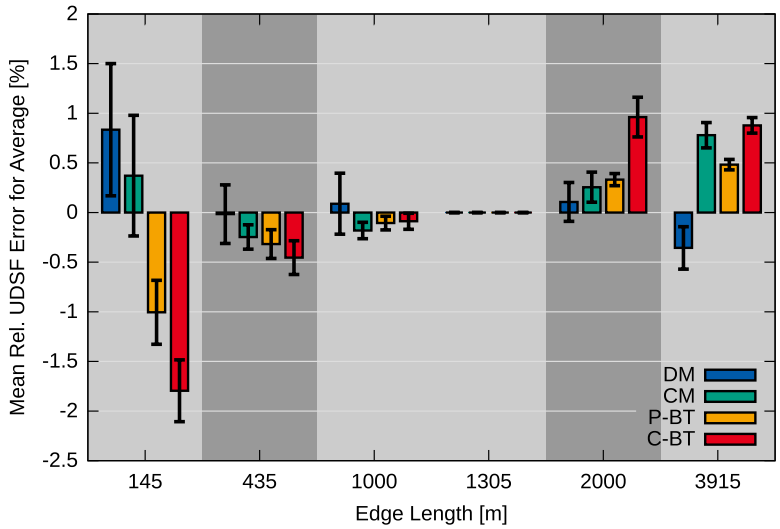
Figure 59: Mean relative node count error for a varying spatial network size

lected and disseminated over multiple levels of the underlying hierarchy. In these scenarios a longer timeout avoids a premature deletion of data from the different tables. Contrary, the prolonged timeout factor leads to an overestimation in networks of a smaller spatial size. It becomes particularly apparent for a spatial network size of 435 m ∗ 435 m, where the underlying hierarchy has a maximum level of one and comes up to a mean relative error of 5.65 %. For the same reason C-BlockTree.KOM attains a comparable mean relative error of 5.09 %, which decreases for the scenarios with an edge length of 1305 m and 3915 m, where the underlying hierarchy of C-BlockTree.KOM is properly established. In contrast, the negative impact of networks with an arbitrarily chosen size is reflected by the increased mean relative error, where the underlying hierarchy is not properly established. For an edge length of 1000 m Figure 58 shows that the concentrating block at l_2 is in the modeled area. However, five concentrating blocks at l_1 are not covered by the modeled area and not populated with nodes. As a result data from nodes in the corresponding sectors are lost, because the data are neither concentrated at the responsible blocks nor sent to the concentrating block at l_2. In terms of an edge length of 2000 m, the responsible block at l_3 is only partially on the map, as depicted in Figure 58. Dependent on the distribution of nodes in the area, it might happen that no or only a small fraction of nodes resides in that block to concentrate data and disseminate the results. Consequently, the data from the surrounding sectors on the map cannot be concentrated and get lost, as reflected by a mean relative error of 59.45 %. In this regard P-BlockTree.KOM benefits (i) from the dissemination of results among sibling sectors at a certain level and (ii) from the prevention of concentrating data at dedicated single blocks. As a consequence the spatial size of the network does not influence the establishment of P-BlockTree.KOM's hierarchy and leads to the depicted small relative mean error.

Both variants of Mobi-G.KOM completely avoid the creation of a hierarchy, thus, the effective spatial size of the network does not influence the accuracy with respect to the node count. As a result both variants of Mobi-G.KOM provide accurate results, which oscillate around a mean relative error of 2 % and outperform both approaches of BlockTree.KOM. However, the outcome for a spatial network size of 3915 m ∗



(a) Mean relative UDSF error using the average as aggregation function

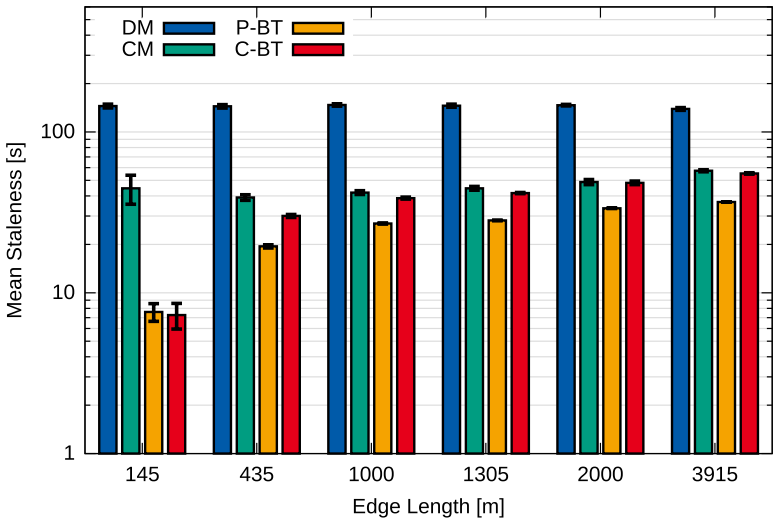


(b) Difference between the varied and default configuration of the scenario parameter regarding the mean relative UDSF error

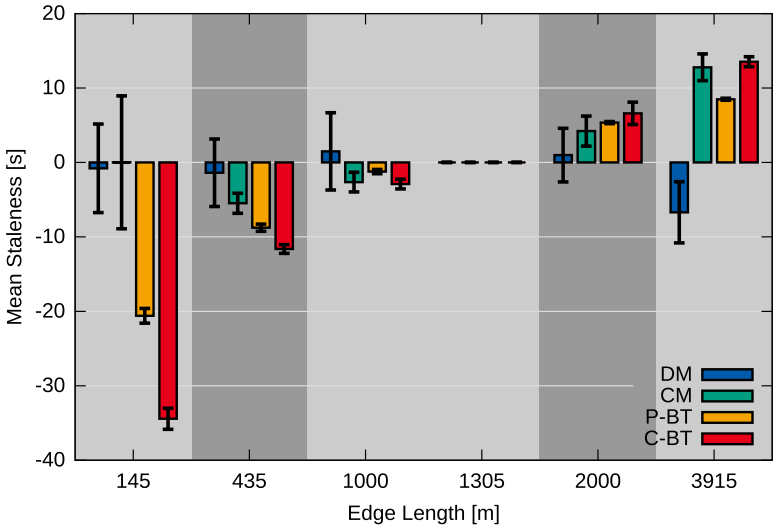
Figure 60: Overview on the mean relative UDSF error for a varying spatial network size

3915 m indicates that the two flat approaches do not perform well in networks of a larger spatial size. The flat structure paired with an increased network diameter impedes the identification and addressing of the beacon as well as the collection of tokens along the paths towards the beacon. As a result a larger fraction of the monitoring results underestimates the current state, as reflected by the increased mean relative error. Contrary to this outcome, the results for BlockTree.KOM reveal that a hierarchical structure counteracts the problems of a large spatial network size with a long network diameter. The underlying hierarchy structures the nodes for the streamlined collection and dissemination of information in large spatial networks.

Figure 60a and Figure 61a unveil the negative influence of a growing spatial network size on accuracy in terms of UDSF as well as on result staleness. Since a larger distance between the nodes prolongs the time to collect and disseminate information, as indicated by the increasing staleness, the mean relative UDSF error increases as



(a) Mean staleness of the provided monitoring results

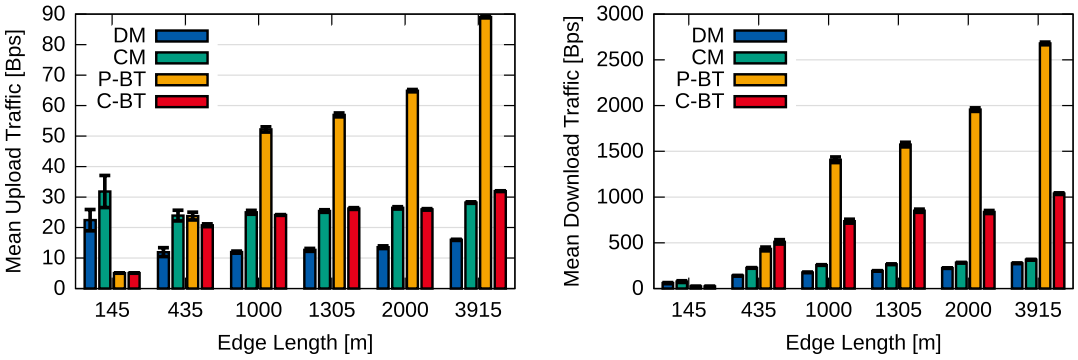


(b) Difference between the varied and default configuration of the scenario parameter regarding the result staleness

Figure 61: Overview on the mean result staleness for a varying spatial network size

well. In this context Mobi-G.KOM DM constitutes an exception, on which we will focus afterwards.

In the following we examine for BlockTree.KOM how a larger spatial network affects both hierarchical approaches. Therefore, we take the default experiment with an edge length of 1305 m as reference, and calculate the difference between the remaining experiments and the reference for the mean relative UDSF error and result staleness (cf. Figure 60b and 61b). The zones with the different shades of gray indicate the extension of the respective hierarchy with another level, ranging from l_0 on the left-hand side to l_3 on the right-hand side. In general, the influence of an increasing spatial network size on both hierarchical approaches is reflected by a stepwise increase. Whenever the hierarchy of P- or C-BlockTree.KOM covers the whole MANET without adding a new level, the impact on accuracy and freshness is rather small. However, if a new level must be added, the new level decelerates the overall



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 62: Overview on the overall traffic per node for a varying spatial network size

collection and dissemination, which leads to the depicted increased mean relative UDSF error and staleness between the differently shaded areas.

In this regard Mobi-G.KOM CM exhibits similar tendencies, as both metrics gradually decrease for smaller areas and increase for larger areas, taking the default experiment as reference (cf. Figure 60b and 61b). The negative impact of larger areas does not result from the addition of a supplementary level, but from longer distances between the nodes to determine the beacon and collect the tokens along the path towards the beacon. The negative influence of longer distances becomes particularly apparent for continuous monitoring, where tokens are collected during an epoch. In terms of Mobi-G.KOM DM, which operates on data from the beginning of an epoch, the effect of longer distances is not that explicit, because the data is ideally collected once at the beginning of an epoch. This particularity becomes apparent by the nearly constant accuracy and freshness, which does not significantly change at least during four of the six experiments. However, the results for the spatial network size of $3915\,\mathrm{m} * 3915\,\mathrm{m}$ indicate that the singular collection of tokens at the beginning of an epoch also suffers from large spatial networks. It is more likely that tokens get lost along the path towards the beacon. This loss has a positive impact on the mean relative UDSF error and staleness, however, the outcome for the monitored number of nodes indicates the effective negative influence.

TRAFFIC AND POWER CONSUMPTION

Figure 62 provides an overview about the arising upload and download traffic of the different monitoring mechanisms and highlights the influence of a varied spatial network size. Figure 62a outlines that both approaches of BlockTree.KOM benefit from smaller scenarios, where the establishment of a hierarchy with multiple levels is avoided. In these scenarios the transmitted amount of data is even smaller than for both variants of Mobi-G.KOM or at least comparable to the resulting upload traffic of Mobi-G.KOM CM. Even for larger scenario, C-BlockTree.KOM still competes with Mobi-G.KOM CM, because both approaches transmit a comparable amount of data. The hierarchical approach benefits from the two facts (i) that the collected data are not disseminated to large areas but concentrated at single blocks and (ii) that a block just belongs to one instead of multiple levels and triggers operations for that level. In terms of P-BlockTree.KOM, the results confirm that the participation of a node at multiple levels leads to a high amount of transmitted data, which increases

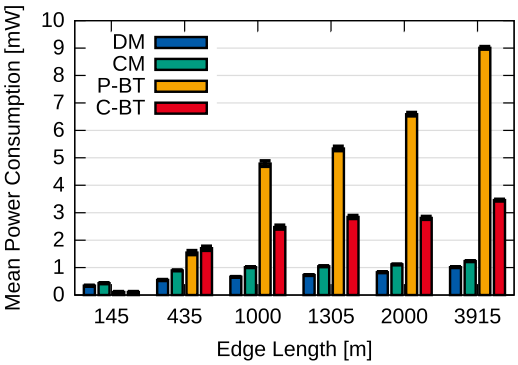


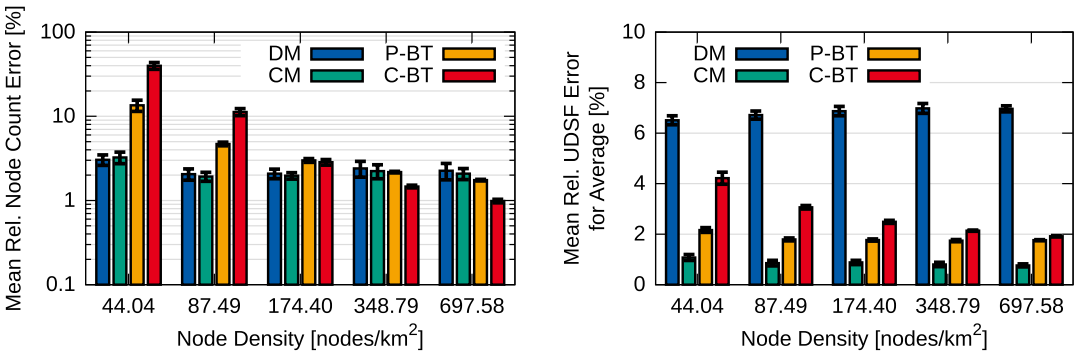
Figure 63: Mean power consumption per node for a varying spatial network size

for larger hierarchies. In this context the two variants of Mobi-G.KOM benefit from their flat design, because no supplementary levels must be added to cover a larger area.

Focusing on the overall mean download traffic per node, Figure 62b unveils that the mean download traffic of all four approaches increases as a function of the spatial network size. However, the results indicate that the differences between the flat and hierarchical approaches increase for larger populated areas. P-BlockTree.KOM suffers from the participation of a node at every active level as well as from the excessive utilization of the receiver-based contention schemes. As already indicated by the mean upload traffic the resulting mean download traffic of P-BlockTree.KOM reflects the negative impact of the combination of both aspects. The influence of the utilized communication schemes becomes also apparent with respect to C-BlockTree.KOM, which competes with Mobi-G.KOM CM in terms of the mean upload traffic but exceeds the flat approach in terms of the mean download traffic.

Based on the separate findings for the mean upload and download traffic, it becomes apparent that the influence of the spatial size of a MANET heavily differs between the hierarchical and flat monitoring mechanisms. Whereas the spatial network size just slightly affects the arising traffic of both variants of Mobi-G.KOM, the results indicate the strong impact on the hierarchical approaches, specifically on P-BlockTree.KOM. This impact originates from the design of both hierarchical approaches and the fact that a larger spatial network size affects every node, which executes a set of operations per level. If the number of levels increases to monitor the whole area, the executed operations multiply, as reflected by the increased traffic. In contrast, the influence of a growing spatial network size mitigates for the flat approaches, because larger or smaller areas do not necessarily affect every node in the network or vary the number of executed operations per node. The design of data collection in Mobi-G.KOM basically affects nodes along the paths towards the beacon, which must forward an increased amount of data. Since aggregation is used to reduce the size of the data along the path, the influence of an increased or reduced number of nodes on the resulting traffic for data collection mitigates. Furthermore, the dissemination of results is triggered by the refresh timeout parameter, which does neither depend on the spatial network size nor on any hierarchy level.

The observed findings are also reflected in Figure 63, which depicts the mean power consumption per node. The metric summarizes the results from the mean upload and download traffic, because both the transmission and the reception of



(a) Mean relative node count error

(b) Mean relative UDSF error using the average as aggregation function

Figure 64: Overview on the influence of a varied node density on the accuracy of the evaluated monitoring mechanisms for a varying node density

data affect the power consumption of a mobile node. Consequently, the arising cost in terms of the mean power consumption underpin and reflect the made observation that C- and specifically P-BlockTree.KOM put a heavy burden on the mobile nodes. This load stems from the hierarchical design paired with the utilized communication schemes and the fact that every node is affected by a larger spatial network.

6.4.1.2 *Variation of the Node Density*

For the assessment of the impact of node density on a decentralized monitoring mechanism we vary the number of nodes and keep the spatial size fixed. Taking the default scenario for the system parameter evaluation as basis, the number of nodes is varied, as depicted in Table 37. Starting from the default configuration with 297 nodes, we double and halve this value twice, which leads to the resulting experiments with the corresponding numbers of nodes and node densities. Based on these densities, we examine and compare performance and cost in sparsely as well as densely populated scenarios.

Scenario parameter	**Configuration**				
Number of nodes	75	149	297	594	1188
Node density [$^{\text{nodes}}/_{\text{km}^2}$]	44.04	87.49	174.40	348.79	697.58

Table 37: Parameter variation of the number of nodes with the specification of the resulting node density for a square area with an edge length of 1305 m

ACCURACY AND STALENESS

The results in Figure 64 immediately outline the negative influence of sparsely populated areas on both hierarchical approaches. For the two experiments, which exhibit a lower node density than the default experiment, the monitoring accuracy degrades, as illustrated in Figure 64a. Even in terms of UDSF (cf. Figure 64b), where the aggregation of data mitigates the influence of missing data, the increasing mean relative error indicates the impact of sparsely populated areas on P- and C-BlockTree.KOM's performance. Two causes are identified that explain the bad performance of P- and

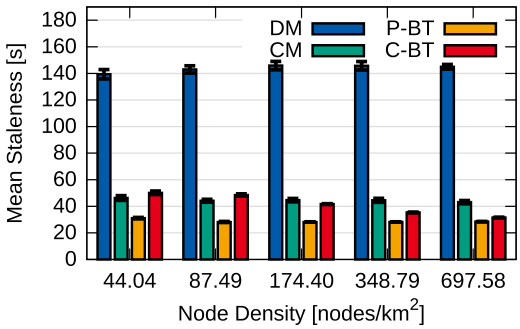
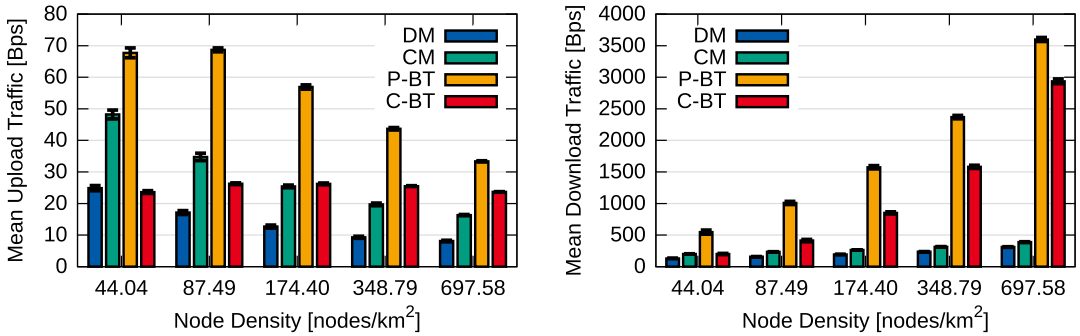


Figure 65: Mean staleness of the provided monitoring results for a varying node density

especially of C-BlockTree.KOM. On the one hand, both approaches require that a path to a targeted concentrating block or to a sector exists so that data are processed and forwarded. With a decreasing density even parallel forwarding of data from multiple nodes, relying on the contention-based forwarding schemes, does not suffice to reach the addressed block or sector. On the other hand, addressed blocks or sectors might be reachable but empty, because no nodes currently reside in that region. Specifically in C-BlockTree.KOM, where data are concentrated at single blocks, the probability of empty or sparsely populated blocks increases so that data get lost. In terms of P-BlockTree.KOM, this probability decreases, because the approach relies on federations of sectors instead of single blocks to process and forward data. Contrary to these observations, the experiments with a higher node density than in the default experiment indicate that the hierarchical approaches benefit from densely populated scenarios. The monitoring accuracy increases and BlockTree.KOM outperforms both variants of Mobi-G.KOM regarding the node count. Specifically C-BlockTree.KOM profits from the increasing density, which ensures a direct and fast transmission of data towards the concentrating blocks, as indicated by the good accuracy as well as reduced staleness of results (cf. Figure 65).

For both variants of Mobi-G.KOM the results for the mean relative error and staleness in Figure 64 and 65 reveal the minor impact of node density on the overall performance. Even in sparsely populated scenarios Mobi-G.KOM is able to provide accurate monitoring results, because the periodic exchange of army messages helps to identify a path towards the beacon as long as this path exists. Once, the monitoring data are collected, the periodic dissemination ensures that the results are distributed among the remaining nodes in the MANET. The results for the mean staleness, which range between 139.38 s and 145.86 s for Mobi-G.KOM DM and between 43.16 s and 46.25 s for Mobi-G.KOM CM, also indicate in this context that the node density does neither degrade nor improve the time to collect and disseminate the information.

The obtained findings outline that P- and C-BlockTree.KOM are sensitive to the node density in a MANET. Whereas both approaches benefit from densely populated areas, the results also indicate that they suffer from sparsely populated networks due to the two main reasons, as identified and discussed above. In this regard the node density in a MANET just slightly influences the flat variants of Mobi-G.KOM, which provide a nearly constant performance independent of the number of nodes per square kilometer. Consequently, they outperform the hierarchical approaches in sparsely populated areas but cannot take an advantage of denser populated areas.



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 66: Overview on the overall mean traffic per node for a varying node density

TRAFFIC

The results for traffic in Figure 66 reveal the effect of a varying node density on the effective mean upload and download traffic per node of all four monitoring mechanisms. As already indicated during the initial examination of BlockTree.KOM and Mobi-G.KOM (cf. Section 6.3.2.2 and 6.3.3.2) the varying node density paired with the utilized contention-based forwarding schemes has an opposed impact on the arising upload and download traffic. Ignoring the experiment with a node density of 44.04 $^{\text{nodes}}/_{\text{km}^2}$ for both approaches of BlockTree.KOM, as this density does not suffice for the proper exchange of information, Figure 66a outlines that the mean upload traffic per node decreases, whereas the mean download traffic per node increases (cf. Figure 66b) as a function of the node density. As already sketched in Section 6.3.2.2 and 6.3.3.2 this interesting effect mainly attributes to the utilization of contention-based forwarding schemes, which are responsible for a large part of the overall upload and download traffic of all mechanisms. A higher density leads to a larger fraction of potential candidates for the initial transmission as well as forwarding of messages, while only a small set of nodes effectively sends and forwards these messages. Due to the higher ratio between potential and effective candidates the mean upload traffic per node decreases. However, the decreasing upload traffic does not compensate for the increased number of nodes in the MANET, which leads to the augmented mean download traffic per node. In addition, the higher node density increases the probability that nodes receive new or redundant messages, amplifying the influence on the mean download traffic per node. Especially BlockTree.KOM, which exclusively relies on contention-based forwarding schemes, suffers from this effect, as indicated by the boosted mean download traffic in Figure 66b. A possible countermeasure for this negative impact is the reduction of potential candidates for the initial transmission and forwarding of messages. This may be achieved by a reduction of the area for the identification of candidates or by preselecting candidates based on the application of a probability function.

With regard to the results for performance and cost we observe that the improved performance of P- and C-BlockTree.KOM in densely populated scenarios comes at the expense of a considerably increased mean download traffic due to the excessive utilization of contention-based forwarding schemes. In contrast, the slight impact of node density on the performance of Mobi-G.KOM DM and CM is also observable

in terms of cost. A reason for the minor impact originates, for instance, from the utilization of unicast instead of broadcast to collect data.

6.4.1.3 *Summary on Scalability*

Before we focus on scalability and summarize our findings from this section, we start with an initial comparison between the hierarchical and flat monitoring mechanisms, relying on the experiment with the default scenario as basis. In general, the results reveal that both variants of Mobi-G.KOM outperform P- and C-BlockTree.KOM regarding accuracy. They provide a more accurate view of the current number of nodes in the network and, in terms of Mobi-G.KOM CM, of the UDSF attribute as well. In contrast, BlockTree.KOM requires the same amount or even less time to provide the participating nodes with the monitoring results. However, BlockTree.KOM cannot derive a benefit from the fast provisioning of results, as reflected by the lower accuracy. This leads to the conclusion that the underlying hierarchy helps to accelerate the collection and dissemination, but affects the accuracy of the provided results in parallel. On this basis, the arising cost reveal that the improved accuracy of Mobi-G.KOM comes at considerably lower cost, as reflected by the lower mean traffic and power consumption per node. The utilization of a hierarchy and the excessive communication due to the exclusive utilization of contention-based forwarding schemes are accountable for the high cost of C-BlockTree.KOM and specifically of P-BlockTree.KOM.

Dealing with the impact of the spatial network size and node density, the results in Section 6.4.1.1 and 6.4.1.2 unveil that both scenario parameters exert a stronger influence on performance and cost of P- and C-BlockTree.KOM than on Mobi-G.KOM DM and CM. Taking the results for the mean relative node count error as reference, P- and C-BlockTree.KOM benefit from the hierarchical topology. The underlying tree of blocks proves to be successful in networks of larger spatial sizes, because it helps to structure the nodes and organizes the data exchange between them. However, the results unveil that C-BlockTree.KOM does not perform well in networks of an arbitrary spatial size. C-BlockTree.KOM requires a fully established hierarchy, where the concentrating blocks are populated with nodes and reachable from the surrounding blocks and sectors. If these conditions are not met, C-BlockTree.KOM cannot provide accurate monitoring results. In this context, P-BlockTree.KOM benefits from the data exchange between sibling sectors instead of the exchange between parent and child blocks. Based on the collection and processing of data in federations of sectors and based on the assignment of a block to all active levels, the planar approach operates in networks of any spatial size. Taking the outcome for UDSF and result staleness into consideration, the hierarchical topology cannot counteract the effect of a larger spatial network size. With a growing hierarchy to cover the MANET the average time to collect and disseminate information increases. Due to the strong relation between the result staleness and the UDSF attribute the mean relative UDSF error increases as well.

Contrary to BlockTree.KOM, the influence of the spatial network size is not that strong on Mobi-G.KOM, because it does neither influence the underlying topology nor the information exchange between the nodes. Consequently, Mobi-G.KOM successfully operates on networks of any spatial size: the mean relative node count error remains nearly constant and the effect on the relative UDSF error and on the result

staleness is comparable to the outcome for P- and C-BlockTree.KOM. However, the outcome for an area of 3915 m * 3915 m indicates that the accuracy of the two flat variants degrades in very large areas. Without a hierarchy to structure the nodes and the data exchange the flat topology suffers from an increased network diameter. Paired with an increased network diameter the flat topology complicates and decelerates the identification of a beacon as well as the collection and dissemination of data. Dealing with the influence on cost, Mobi-G.KOM benefits from the fact that the size of the monitored area does neither influence the information exchange nor the flat topology. Moreover, both variants avoid the extensive and exclusive utilization of contention-based forwarding schemes and rely on unicast for the collection of tokens. Consequently, the impact on the mean upload and download traffic as well as on the resulting mean power consumption mitigates.

The results for a varying node density exhibit similar tendencies as for the variation of the spatial network size: performance and cost of BlockTree.KOM depend on the varied scenario parameter, whereas the effect mitigates for Mobi-G.KOM. Taking the results for the node count as reference, the accuracy of P-BlockTree.KOM and specifically of C-BlockTree.KOM degrades. Again, P-BlockTree.KOM benefits from its federations of sectors, because nodes from multiple blocks are in charge to collect and process the data, increasing the probability that the monitoring data reach their destination and are processed by present nodes at the destination. C-BlockTree.KOM relies on single blocks, which must collect and process the data from the surrounding blocks and sectors. If these blocks are not populated or not reachable by the surrounding blocks and sectors, the data are lost, as reflected by the increasing mean relative node count error for sparsely populated MANETs. With a higher density of nodes P- and C-BlockTree.KOM benefit from the underlying hierarchy, as acknowledged by a faster delivery and a higher accuracy of the provided results. In terms of the node count, both approaches even outperform Mobi-G.KOM, because the hierarchy organizes the nodes for a streamlined data collection and result dissemination and successfully handles the higher number of nodes. Similar to the negative influence of larger spatial network sizes on traffic and power consumption the gain in performance comes at higher cost. The considerably increased mean download traffic per node stems from the excessive and exclusive utilization of contention-based forwarding schemes paired with the increased density of nodes. In contrast to BlockTree.KOM the results uncover that both variants of Mobi-G.KOM are impervious to the node density of a MANET. The performance in terms of accuracy and staleness remains nearly constant, whereas the impact on traffic is damped, because Mobi-G.KOM does not exclusively rely on contention-based forwarding schemes to communicate.

6.4.2 *Evaluating Robustness*

We exploit the evaluation of robustness to examine the influence of the inherent characteristics of MANETs on a decentralized monitoring mechanism. The related experiments for the evaluation are divided into three parts. In the first part, which is presented in Section 6.4.2.1, we vary the movement speed of the mobile nodes and examine its impact. Subsequently, Section 6.4.2.2 deals with the evaluation of a decentralized monitoring mechanism under different levels of churn, where we

decrease and increase the mean node session length. Finally, we vary the interval for the GPS utilization and evaluate how the approaches perform with recent or stale position information (cf. Section 6.4.2.3).

6.4.2.1 *Variation of the Movement Speed*

The configuration of the movement speed for the steady-state Random Waypoint Mobility model [120] is configurable by two parameters: the mean movement speed and the maximum deviation from the mean. Given the default scenario configuration for the system parameter evaluation, the mean movement speed is set to 1.5 m/s with a maximum deviation of 0.5 m/s leading to a minimum and maximum movement speed of 1 m/s and 2 m/s, respectively. During the following experiments, we vary the mean movement speed but keep the maximum deviation so that the resulting interval does not change. Table 38 lists the respective configurations for the mean as well as the minimum and maximum movement speed. Based on the chosen configurations, we investigate (i) if the performance improves in rather static scenarios and (ii) if our approaches are capable to monitor highly dynamic nodes with a mean movement speed of 4 m/s or 8 m/s, which exceeds the speed of a normal pedestrian.

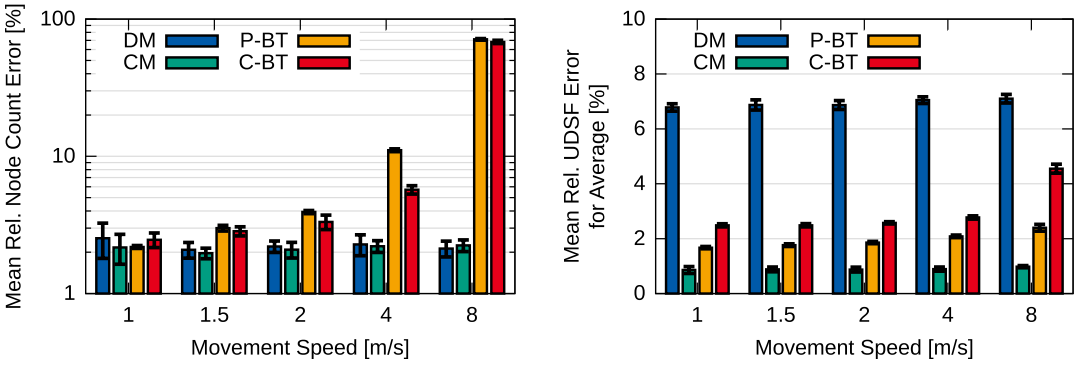
Scenario parameter	**Configuration**				
Minimum movement speed [m/s]	0.5	1	1.5	3.5	7.5
Mean movement speed [m/s]	1	1.5	2	4	8
Maximum movement speed [m/s]	1.5	2	2.5	4.5	8.5

Table 38: Parameter variation of the mean movement speed for the steady-state Random Waypoint Mobility model [120]

ACCURACY

An overview on the influence of the movement speed on accuracy is given in Figure 67. The overview comprises the results for the mean relative node count as well as UDSF error (cf. Figure 67a and 67b), which illustrate a strong impact on P- and C-BlockTree.KOM. Whereas the results for the mean relative UDSF error still conceal the negative influence due to the compensating aggregation function, the results for the node count unveil the existing problems and the strong influence of mobility on BlockTree.KOM. Figure 67a shows that even a reduction of the average movement speed to 1 m/s improves the accuracy, whereas a higher average speed considerably degrades the performance of P- and C-BlockTree.KOM. The primary cause for the degrading accuracy originates from the logical partitioning into blocks paired with the procedure to handle nodes that enter a new block. If a node detects that it entered a new block, it does not actively trigger Aggregating-Up or Disseminating-Down operations for a certain amount of time[5], but just forwards data from other nodes. Based on this behavior, P- and C-BlockTree.KOM avoid that wrong and stale information from other blocks and sectors are mistakenly spread. If the movement speed of the nodes increases, a large fraction of nodes changes the blocks more frequently

5 We refer the interested reader to Appendix A.2.6, where the influence of the corresponding system parameter $t_{blockOperation}$ is examined.



(a) Mean relative node count error

(b) Mean relative UDSF error using the average as aggregation function

Figure 67: Overview on accuracy of the provided monitoring results for a varying mean movement speed

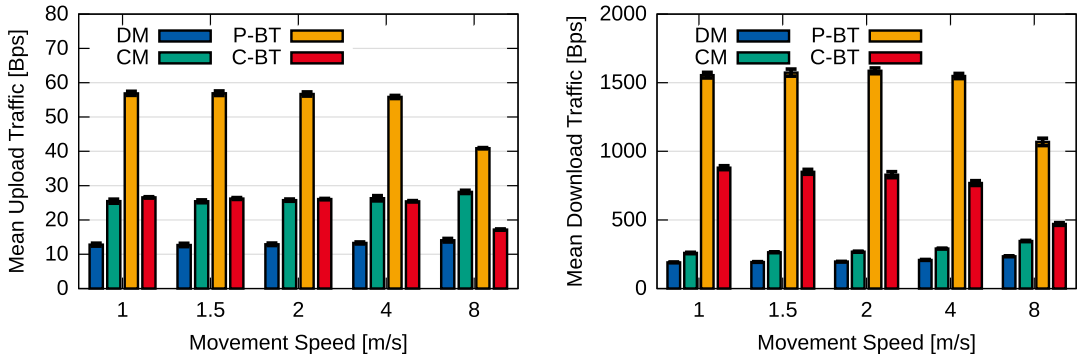
so that the number of nodes, which trigger an AGGREGATING-UP or DISSEMINATING-DOWN operation, decreases. As a result neither the collection of data nor the dissemination of results takes place, as reflected by the increasing mean relative node count and UDSF error. During the discussion of the influence on cost, we further elaborate on this peculiarity.

In terms of MOBI-G.KOM, the results reveal that the movement speed has little influence on the monitoring accuracy of both attributes. Indeed, the results outline that the periodic transmission of army messages suffices even in highly dynamic scenarios to determine the beacon as well as to maintain the different paths towards the beacon. These paths resist the fast changing neighborhood of a node and serve for the successful collection of tokens. Moreover, as MOBI-G.KOM avoids the logical partitioning of the network into blocks and does not ban nodes from the active participation, both variants of MOBI-G.KOM integrate the input from all present nodes in the network. As a result MOBI-G.KOM provides an accurate view of the MANET irrespective of the movement speed of the participating nodes.

The presentation and discussion of the outcome for the mean staleness are omitted, because the results conform to the outcome and findings for the mean relative UDSF error (cf. Figure 67b).

TRAFFIC

Figure 68a and 68b uncover that MOBI-G.KOM's resulting upload and download traffic increase as a function of the movement speed to achieve the observed performance and to handle the increasing dynamics. The growing overall traffic of MOBI-G.KOM DM and CM is attributed to the increasing exchange of IS and army messages. Taking the resulting mean download traffic as representative, Figure 69a depicts the growing army traffic, which is necessary to identify the beacon and, particularly, to maintain the shortest paths towards that beacon in the presence of a constantly changing neighborhood and overall network topology. Similarly, Figure 69b outlines that the IS traffic increases as well to manage the dissemination of the current state over the fast changing topology. In addition the results for the IS traffic reveal that the impact on MOBI-G.KOM CM is stronger than on MOBI-G.KOM DM, because the



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

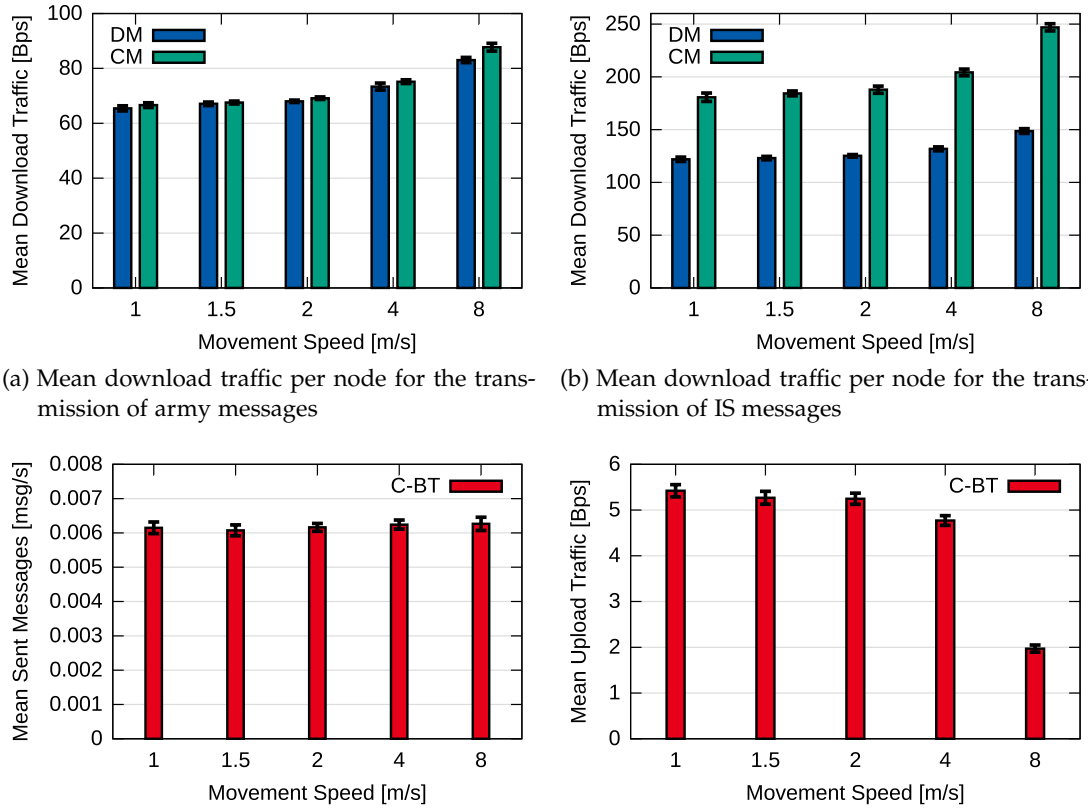
Figure 68: Overview on the overall mean traffic per node for a varying mean movement speed

first variant updates the current state multiple times during an epoch, whereas the latter variant just spreads the monitoring results once per epoch.

Regarding P- and C-BlockTree.KOM, Figure 68a and 68b outline that the upload and download traffic is nearly constant for the first four experiments but collapses for the fifth one. The reason for the bad performance, as previously discussed, as well as for the sudden decrease in traffic is attributed to the blocking of operations for a certain amount of time. To prevent the dissemination of incorrect data a node does not actively participate for this period of time and avoids a transmission of own data, which leads to the degrading performance as well as to the reduced traffic. To underpin this statement we take a closer look at the transmitted replication traffic of C-BlockTree.KOM, which explains the behavior of P-BlockTree.KOM as well. Figure 69c depicts the mean number of sent replication messages, whereas Figure 69d shows the resulting traffic due to the exchange of replication messages. When comparing the results from both figures, we observe that the amount of messages does not change dependent on the movement speed, which indicates that a block is always populated with a comparable amount of nodes that replicate their hierarchy table. However, Figure 69d reveals that the traffic decreases, because the hierarchy table holds no data for replication. This example highlights the fact that the concentrating blocks of C-BlockTree.KOM do not receive the required information from the associated sectors, because most of the nodes are currently blocked, since they entered a new block. Due to this blocking of nodes, which leads to the lack of information, P-BlockTree.KOM's overall upload and download traffic collapses as well.

6.4.2.2 *Variation of Mean Node Session Length*

To study the impact of shorter and longer session times on decentralized monitoring mechanisms we vary the scenario parameter for the mean node session length. Table 39 depicts the corresponding configurations for the scenario parameter. During the experiments, we examine as well the influence of a longer session length on performance and cost of a decentralized monitoring mechanism. However, the focus is on the evaluation of shorter session lengths below the default configuration to investigate if a decentralized monitoring mechanism is able to integrate or delete information from nodes with short session lengths.



(a) Mean download traffic per node for the transmission of army messages

(b) Mean download traffic per node for the transmission of IS messages

(c) Mean number of transmitted replication messages

(d) Mean upload traffic per node for data replication

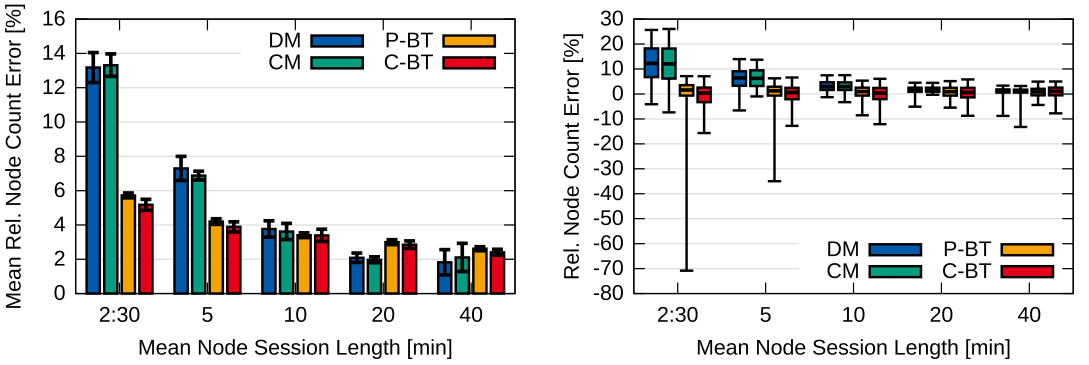
Figure 69: Overview on the mean upload and download traffic per node for the transmission of different types of messages

ACCURACY

To rate the impact of different mean node session lengths on performance we only focus on the mean relative node count error, because the influence on the mean relative UDSF error as well as on the staleness of results is negligible. Figure 70a depicts the mean relative node count error as a function of the mean node session length. Overall, the results reveal that a decreasing session length influences the accuracy of all four decentralized monitoring mechanisms, as illustrated by the increasing relative error. Even a longer mean node session length than for the default configuration still influences accuracy, which is reflected by a smaller error in terms of P- and C-BlockTree.KOM and by the varying error with respect to Mobi-G.KOM DM and CM. For shorter session lengths the results unveil the problems of both variants of Mobi-G.KOM with churn, which leads to the increased mean relative error and a degrading performance compared to BlockTree.KOM. As already discussed in Section 6.3.3.2, where we examined the influence of scenarios with and without churn,

Scenario parameter	**Configuration**				
Mean node session length [min]	2:30	5	10	20	40

Table 39: Parameter variation of the mean node session length



(a) Mean relative node count error

(b) Distribution of the relative node count error of all collected measurements

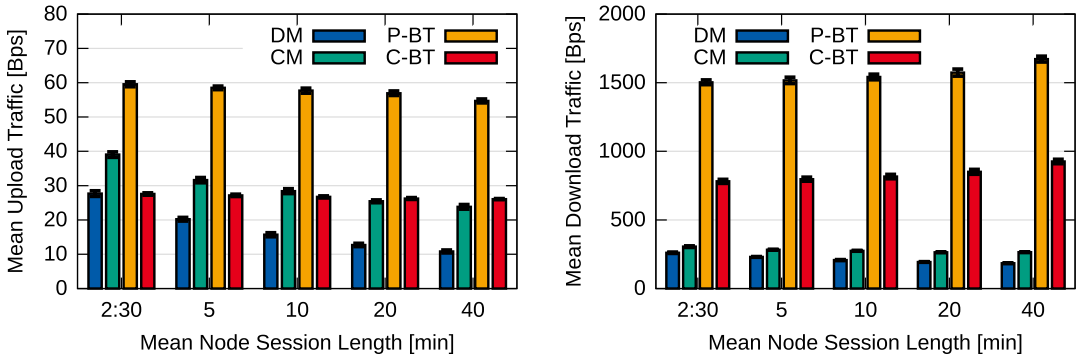
Figure 70: Overview on the mean relative node count error for a varying mean node session length

both variants of MOBI-G.KOM suffer from the transmission of old tokens from nodes, which already left the network. In terms of the node count, MOBI-G.KOM DM and CM only prune the current view of the network state from old data by restarting the epoch. Up to that point in time, old information is always reflected in the global view of an attribute. With a decreasing mean session length the amount of old information increases, because more nodes leave the network during the epoch, while the data is still integrated. Consequently, the growing mean relative error originates from a growing overestimation for shorter session lengths, as depicted in Figure 70b, which illustrates the distribution of the relative node count error of all collected measurements. As opposed to the increasing overestimation, the results for BLOCKTREE.KOM uncover that both approaches suffer from a growing underestimation for shorter node session lengths. In terms of C-BLOCKTREE.KOM, the majority of obtained monitoring results tends to underestimate the number of nodes in the network. In contrast, P-BLOCKTREE.KOM suffers from a smaller fraction of inaccurate monitoring results. However, this fraction considerable underestimates the current state.

TRAFFIC

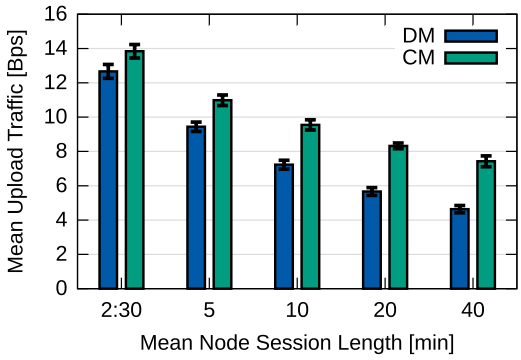
With respect to cost Figure 71a and 71b depict the mean upload and download traffic per node as a function of the mean node session length. In terms of P- and C-BLOCKTREE.KOM, the previously discussed effect (cf. Section 6.4.1.2) of a varying node density paired with the contention-based forwarding schemes is observable, because the mean upload traffic per node decreases, whereas the mean download traffic increases. With a longer mean node session length the number of active nodes slightly increases accompanied by a higher node density that explains the decreasing mean upload and increasing mean download traffic. Apart from this effect, a varied mean node session length has no further influence on BLOCKTREE.KOM's resulting traffic, because both approaches do not define specific methods to handle arriving or leaving nodes.

Contrary to this outcome, the results for MOBI-G.KOM unveil that both variants are influenced by a varying mean node session length, because the mean upload and even the mean download traffic increase for shorter mean session times. Due to the large and unfavorable scaling in Figure 71b the increased mean download

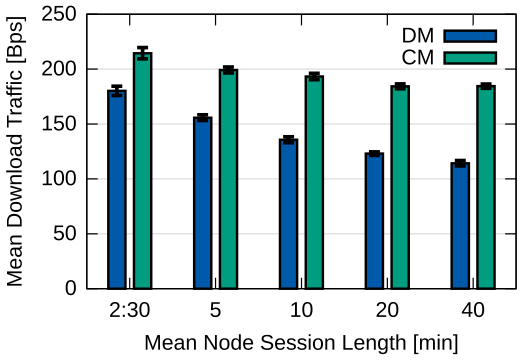


(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node



(c) Mean upload traffic per node for the transmission of IS messages



(d) Mean download traffic per node for the transmission of IS messages

Figure 71: Overview on the mean overall traffic as well as on the mean IS traffic per node for a varying mean node session length

traffic is hardly observable, but increases from 184.26 Bps to 260.92 Bps in terms of Mobi-G.KOM DM and from 264.66 Bps to 305.34 Bps for Mobi-G.KOM CM. We take the resulting IS traffic as attempt to explain this behavior, because it accounts for the majority of the overall traffic. The results in Figure 71c and 71d reveal that a reduced mean node session length leads to an increased mean upload and download traffic for the exchange of IS messages. The reason for the growing traffic stems from the fact that the frequency of arriving nodes during an epoch increases as a function of smaller session times. These nodes must be integrated by collecting their locally measured attributes, which leads to a frequent update of the current network state during an epoch. Consequently, IS messages are spread multiple times during an epoch to reach a steady state.

6.4.2.3 *Variation of the GPS Utilization*

To complete the evaluation regarding robustness we vary the interval for the periodic utilization of GPS and assess how our decentralized monitoring mechanisms operate with recent as well as stale position information. Table 40 lists the corresponding configurations for the position refresh interval, which determines the length for the periodic retrieval of position information from the GPS receiver.

Scenario parameter	Configuration				
Position refresh interval [s]	5	10	20	40	80

Table 40: Parameter variation of the position refresh interval

ACCURACY

For the assessment of the impact of a varying position refresh interval we limit our investigations on the monitoring accuracy of the node count. The variation of this parameter has little influence on the mean relative UDSF error as well as on staleness. Figure 72 depicts the mean relative node count error as a function of the position refresh interval. With respect to both variants of MOBI-G.KOM the results reveal that a variation of this scenario parameter has little influence on the accuracy of both variants. For all configurations of the position refresh interval MOBI-G.KOM DM and CM attain a mean relative node count error around two percent. The reason for MOBI-G.KOM's good performance despite the varied parameter originates from the fact that both variants require position information to calculate the hesitation time for forwarding IS and army messages or for responding on incorrect or stale data. Consequently, only MOBI-G.KOM's communication is affected, as discussed below, which does neither influence accuracy nor the timeliness of results.

With respect to P- and C-BLOCKTREE.KOM the results in Figure 72 unveil that a varied position refresh interval influences the accuracy in terms of the node count. In the first four experiments C-BLOCKTREE.KOM attains a mean relative error around three percent, whereas the mean relative error of P-BLOCKTREE.KOM even decreases for a longer refresh interval. This interesting effect is attributable to the reduced frequency of banning nodes from triggering AGGREGATING-UP operations, because it takes longer to detect that a node entered a new block. During the prolonged detection of the current position, nodes actively participate in P-BLOCKTREE.KOM and do not limit their contributions to forwarding data from other nodes. As a result monitoring data from these nodes are still integrated so that P-BLOCKTREE.KOM does not suffer from missing information due to blocked nodes, as reflected by the results in Section 6.4.2.1, where the high mobility leads to a large fraction of blocked nodes and a bad performance. At the same time, a node rejects received data from other nodes of foreign blocks and vice versa so that information from different blocks is not mixed. However, for a position refresh interval of 80 s, the accuracy of both hier-

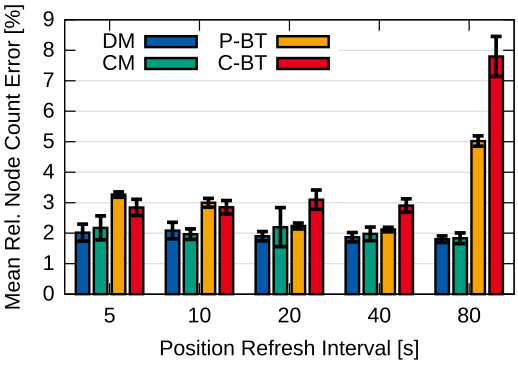
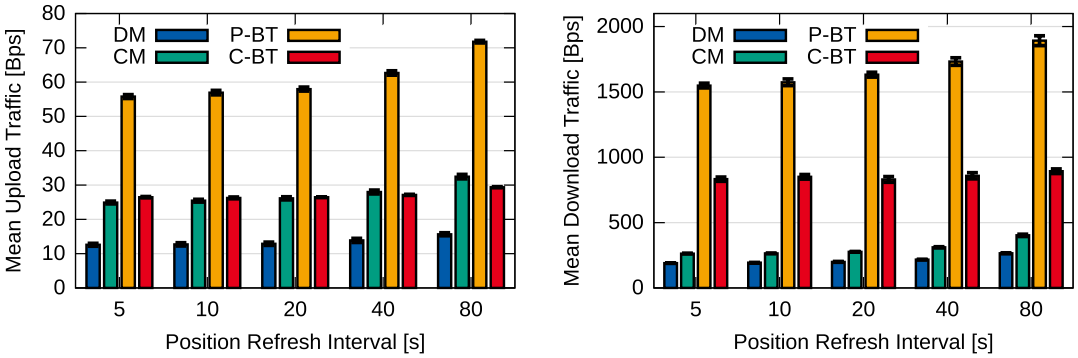


Figure 72: Mean relative node count error for a varying position refresh interval



(a) Mean overall upload traffic per node

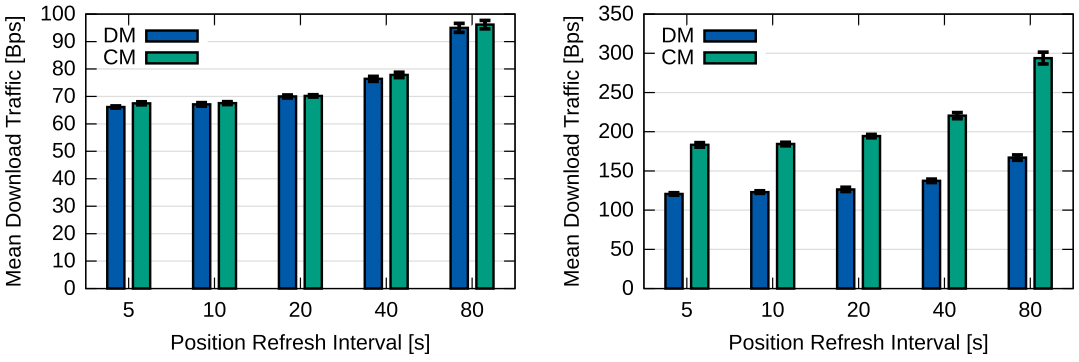
(b) Mean overall download traffic per node

Figure 73: Overview on the overall mean traffic per node for a varying position refresh interval

archical approaches degrades, because the previously sketched procedure does not suffice to compensate very old information about a node's current position. Contrary to Mobi-G.KOM, position information is not only required for the contention-based forwarding schemes but to determine a node's current block and level in the tree of blocks. With a longer position refresh interval the deviation between the calculated and effective level and block grows, as illustrated by the decreasing accuracy in terms of the node count.

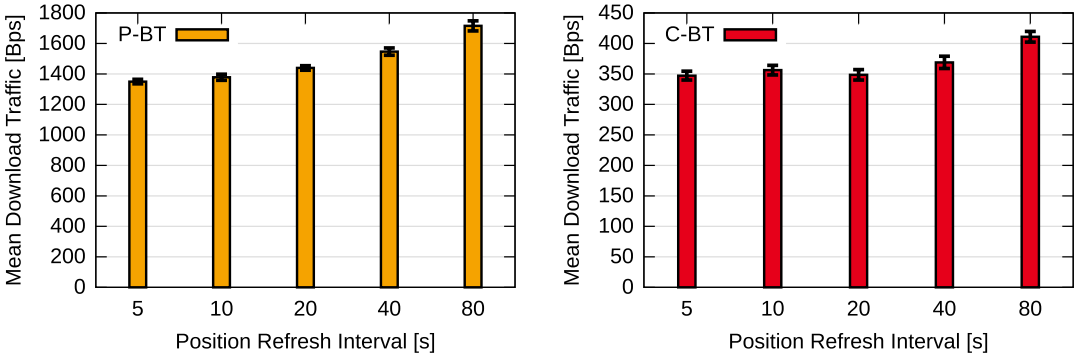
TRAFFIC

The resulting upload and download traffic of the four monitoring mechanisms is depicted in Figure 73a and 73b, respectively. Based on the outcome, it becomes apparent that the upload and download traffic increases with a larger position refresh interval. The main reason for the overall increase originates from the utilization of the contention-based forwarding schemes paired with the inaccurate information about a node's current position. Due to stale and wrong information messages are forwarded by nodes, which, for instance, do not maximize the distance to the previous hop and increase the distance to the destination in the worst case. As previously discussed for Mobi-G.KOM this peculiarity has no influence on performance but leads to growing cost. Taking the download traffic as an example, Figure 74a and 74b depict the growing army and IS download traffic as a function of the longer position refresh interval, which underpins the made observation. In terms of Mobi-G.KOM CM, the stronger influence on the download traffic for the transmission of IS messages results from the continuous integration of tokens and subsequent dissemination of updated results during an epoch. Similarly, the results in Figure 74c and 74d show the same tendency for the download traffic from the execution of Aggregating-Up operations in P- and C-BlockTree.KOM. Furthermore, the reduced amount of banned nodes, which do not actively trigger Aggregating-Up and Disseminating-Down operations, represents another factor for the increased download traffic. This effect has a stronger impact on P- than on C-BlockTree.KOM, because nodes trigger the corresponding operations not only for one but for every active level.



(a) Mean download traffic per node for the transmission of army messages

(b) Mean download traffic per node for the transmission of IS messages



(c) Mean download traffic per node arising from AGGREGATING-UP operations of P-BLOCKTREE.KOM

(d) Mean download traffic per node arising from AGGREGATING-UP operations of C-BLOCKTREE.KOM

Figure 74: Overview on different types of traffic for a varying position refresh interval

6.4.2.4 *Summary on Robustness*

Before we recapitulate the impact of the different experiments on MOBI-G.KOM and BLOCKTREE.KOM, we overview the influence of the experiments on performance. The implications on cost are considered during the summary of the respective approach. With respect to performance the obtained results and findings for robustness outline that the challenging characteristics of MANETs, which have been identified as one of the key challenges in Chapter 1, as well as the treatment of inaccurate and stale position information mainly affect accuracy, whereas the impact on timeliness of the results is rather small. Regarding accuracy, the influence of the varied scenario parameters is mainly reflected by the node count attribute. The observable influence mitigates or is concealed, when considering the results for UDSF.

The results from the experiments for robustness indicate that MOBI-G.KOM represents a robust decentralized monitoring mechanism, which handles fast moving nodes and operates on inaccurate and stale position information. In the first case MOBI-G.KOM benefits from its flat design and the fact that the collection of monitoring data and the dissemination of results do not depend on dedicated locations. Instead, the results reveal that the identification of dedicated nodes, which are chosen for a short amount of time to collect and disseminate information, represents a good and robust design decision. Paired with the robust communication pattern of gossiping for (i) the identification and maintenance of a beacon, (ii) the dissemination

of results, and (iii) the collection of tokens over the identified paths Mobi-G.KOM DM and CM handle even fast moving nodes in a way that the identified responsible nodes are always reachable and provide accurate monitoring results. Moreover, the impact of stale position information on performance is negligible, as reflected by the constant accuracy for the node count and by the omitted discussion of results for UDSF and result staleness. The minor impact on Mobi-G.KOM's performance originates from the fact that position information is just utilized by the contention-based forwarding schemes to exchange information. However, the results for cost unveil that the utilization of stale position information leads to an increased traffic, because the contention-based forwarding schemes do not select appropriate nodes to forward IS and army messages. Consequently, more messages must be transmitted, leading to an increasing upload and download traffic. A similar tendency is observable for the experiments, where the movement speed of the nodes is varied. Mobi-G.KOM pays for its good performance with an increased overall traffic to handle fast moving nodes. The traffic increases, because the fast changing topology complicates the identification and maintenance of a beacon as well as the dissemination of results. Likewise, the results for a varied mean node session length indicate that the traffic grows if Mobi-G.KOM has to deal with short lived-nodes. However and contrary to the other experiments, Mobi-G.KOM exhibits problems to handle this type of dynamic, which arises from the session length of a node. The outcome for short-lived nodes unveils that the division of time into epochs and the synchronization of nodes according to these epochs leads to a degrading performance. Due to the time-based organization of both variants the current view of the network in terms of the node count can only be corrected by a restart of the epoch. The more nodes leave during an epoch, the stronger is the negative impact on Mobi-G.KOM's performance.

In contrast, BlockTree.KOM does not rely on a time-dependent organization with a periodical restart of all operations, but specifies operations dependent on locations. Consequently, P- and C-BlockTree.KOM provide an accurate view in the presence of short-lived nodes. However, the findings outline that the dependence on locations is highly sensitive to the mobility of nodes and also affected by the freshness of position information. Despite the creation of a hierarchy, which is established on top of relations between blocks with multiple nodes instead of relations between single nodes, the resulting tree of blocks cannot handle fast moving nodes that rapidly cross the underlying blocks. As a consequence the accuracy of the provided results decreases with an increasing movement speed and even collapses for fast moving nodes. In terms of the freshness of position information, the findings indicate that P- and C-BlockTree.KOM handle even stale position information to a certain extent. However, the accuracy of the provided results degrades for very stale information. In contrast to Mobi-G.KOM BlockTree.KOM additionally requires the position information to establish and maintain its tree of blocks. If the information is old and inaccurate, the establishment and maintenance of the hierarchy is affected, complicating the collection and dissemination of information. Considering the effect on BlockTree.KOM's arising cost, a direct impact on traffic is only observable for the experiments with the varied position refresh interval. The results for the varied node session length unveil that the traffic changes. However, this change cannot be attributed to the varied scenario parameter. In fact, the resulting effect on the upload and download traffic originates from the utilized contention-based forwarding schemes paired with the varied density, which has been already discussed during the evaluation of scalability

(cf. Section 6.4.1). In terms of the varied movement speed, the results reveal that the mobility has nearly no impact on the resulting traffic of BLOCKTREE.KOM, except for the last experiment. For very fast moving nodes the traffic collapses, because P- and C-BLOCKTREE.KOM transmit a reduced amount of data due to the increased fraction of blocked nodes.

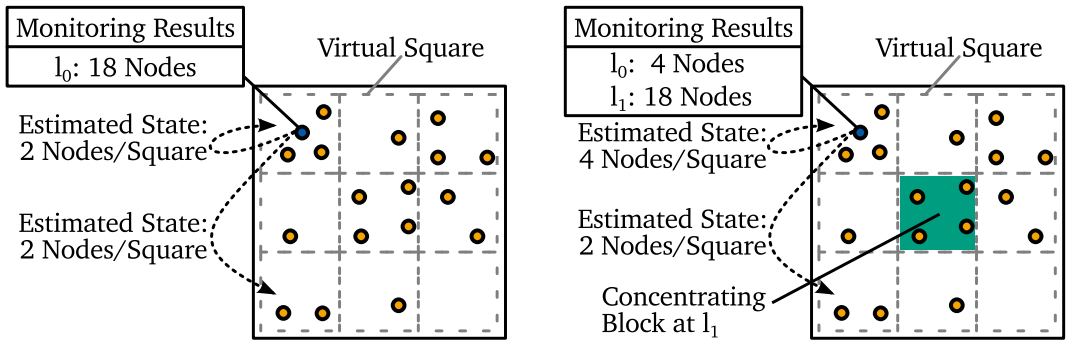
6.4.3 *Evaluating Location-Awareness*

We complete the comparative evaluation by examining the provisioning of location-aware monitoring results. To assess and rate the performance in terms of location-aware monitoring and to evaluate the influence of the different underlying monitoring topologies on the location-aware provisioning of monitoring results we adopt the evaluation procedure from our previous work [157]. The corresponding attributes, which are utilized throughout this evaluation comprise (i) node density, which bases on the node count attribute, and (ii) another attribute, which bases on measurements from the participatory sensing application da_sense, as already introduced in Section 6.2.2. The attribute represents the measured noise level of the surrounding environment, which is locally monitored by every individual node. Based on the data set from da_sense, we simulate the behavior of the attribute to model varying values that depend on time and location of the corresponding measurements. With the utilization of this attribute we extend the application area of MOBI-G.KOM and BLOCKTREE.KOM by the supplementary monitoring of environmental attributes, which do not belong to the category of attributes that characterize or rate the current state of the MANET. However, its utilization during the evaluation represents a descriptive and intuitive example for location-aware monitoring that additionally targets further application scenarios beyond monitoring the current state of participating nodes, the interconnecting links, and end-to-end abstractions, as identified by Kurose and Ross [96].

In the following we describe the scenario design of the executed experiments with a focus on (i) data preparation for the noise level attribute and on (ii) the calculation of the mean relative error to rate the accuracy of location-aware monitoring. Subsequently, we present the results for node density and the noise level.

6.4.3.1 *Experimental Design*

To evaluate the location-awareness of MOBI-G.KOM and BLOCKTREE.KOM we rely on the default scenario from the system parameter evaluation. The scenario is configured as introduced in Section 6.3.1 and the corresponding scenario parameters are set as listed in Table 19. To conduct the evaluation we tile the modeled area into *virtual squares* of a certain size as illustrated by the dashed grey squares in Figure 75. Every node uses its available monitoring results to calculate (i) the node density and (ii) the measured volume of the noise level for every virtual square. In terms of MOBI-G.KOM the participating nodes just rely on the overall global view of the corresponding attribute to estimate the current state in their own as well as the remaining virtual squares. Regarding BLOCKTREE.KOM, the nodes benefit from the different tables that hold information of different granularity to estimate the state of their surrounding virtual square as well as of the remaining ones. Figure 75a and 75b depict an example for MOBI-G.KOM and C-BLOCKTREE.KOM with $l_{max} = 1$. Using the pro-



(a) Example for the calculation of node density for two virtual squares based on the provided monitoring results from Mobi-G.KOM

(b) Example for the calculation of node density for two virtual squares based on the provided monitoring results from C-BlockTree.KOM

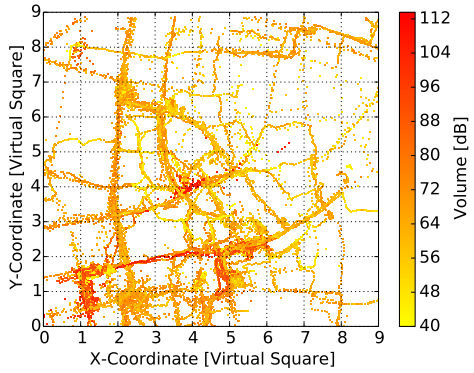
Figure 75: Example of the calculation of node density and generation of location-aware monitoring results

vided monitoring results, a node in the upper left corner calculates the node density for its surrounding and lower left virtual square, as indicated by the dashed black arrows in Figure 75.

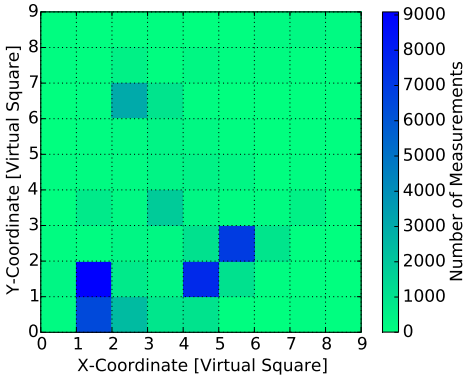
To assess the accuracy of the results the relative error between the monitored aggregate $X_m(A,t,K)$ and the effective aggregate $X_c(A,t,K)$ is calculated for every virtual square in the modeled area. Based on the hierarchy level, which provides the required information to estimate the state of the virtual square, the obtained results are grouped for the subsequent evaluation. The grouping of the results according to the hierarchy level enables (i) to assess if the considered decentralized monitoring mechanism provides location-aware monitoring results and (ii) to evaluate the accuracy of those results. As both variants of Mobi-G.KOM are flat approaches without a hierarchy the results belong to the group for l_0.

The data of the noise level measurements from da_sense must be preprocessed so that they can be monitored during an experiment. For this reason the measurements from different positions are grouped according to the size and position of the created virtual squares. Figure 76a depicts the effective position of every measurement in the city center of Darmstadt, whereas Figure 76b shows the available measurements for every virtual square if the single measurements are clustered and accumulated according to the underlying virtual squares. Based on the clustered measurements, Figure 76c shows the average volume per cluster and Figure 76d depicts its standard deviation to characterize the variability of the measurements per cluster. To model the variations of the attribute over time the retrievable measurements per virtual square change as a function of time if multiple measurements are available for that virtual square. We denote the corresponding parameter for the definition of the interval as *measurement update interval*. If the parameter is not configured, the average per virtual square is used, which leads to an assignment of measurements to the virtual squares, as depicted in Figure 76c.

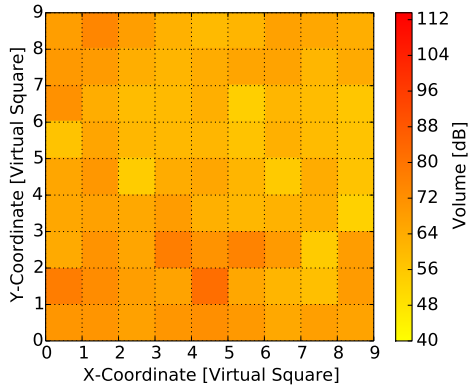
For the following evaluation we set the size of a virtual square to the same size as a block in BlockTree.KOM. Consequently, we obtain nine virtual squares per dimension, which leads to 81 virtual squares, as shown in Figure 76. We evaluate both approaches of BlockTree.KOM and both variants of Mobi-G.KOM, using the respec-



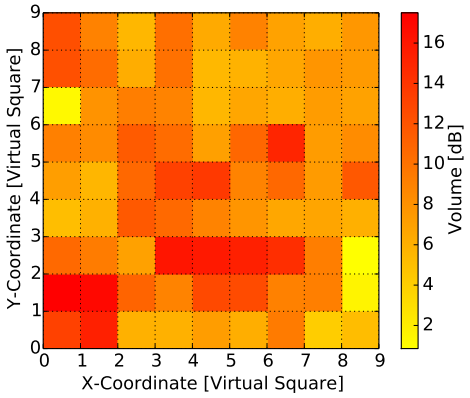
(a) Distribution of the da_sense measurements in the city center of Darmstadt. The colors indicate the volume of the noise level measurements.



(b) Clustering of the da_sense measurements according to the underlying tiling of the area into virtual squares. The colors indicate how many measurements are available per virtual square.



(c) Average volume of the measured noise level per virtual square. The average is calculated based on the available measurements per virtual square.



(d) Standard deviation of the measured noise level per virtual square. The standard deviation is calculated based on the available measurements per virtual square.

Figure 76: Preprocessing of the da_sense data for the utilization as monitoring attribute in an experiment

tive configurations, as described in Section 6.3.2.5 and 6.3.3.6. For P-BLOCKTREE.KOM we additionally set the sector edge length to two so that a sector of a higher level consists of four instead of nine sectors. The configuration of the parameter leads to a hierarchy of five levels with $l_{max} = 4$. Based on this supplementary configuration, we examine the impact of a higher hierarchy on the provisioning of location-aware results. To differentiate between the two configurations of P-BLOCKTREE.KOM we denote the configuration with $l_{max} = 2$ as *PBT-2* and $l_{max} = 4$ as *PBT-4*. In terms of C-BLOCKTREE.KOM we evaluate the approach in two different ways to highlight the impact of the concentrating blocks. We present the results for C-BLOCKTREE.KOM, which are obtained from all participating nodes as well as from the fraction of nodes, which currently reside in concentrating blocks. We denote the outcome based on the results from all nodes as *CBT-A* and term the second one as *CBT-C*, which exclusively comprises the results from the nodes in the concentrating blocks. The measurement

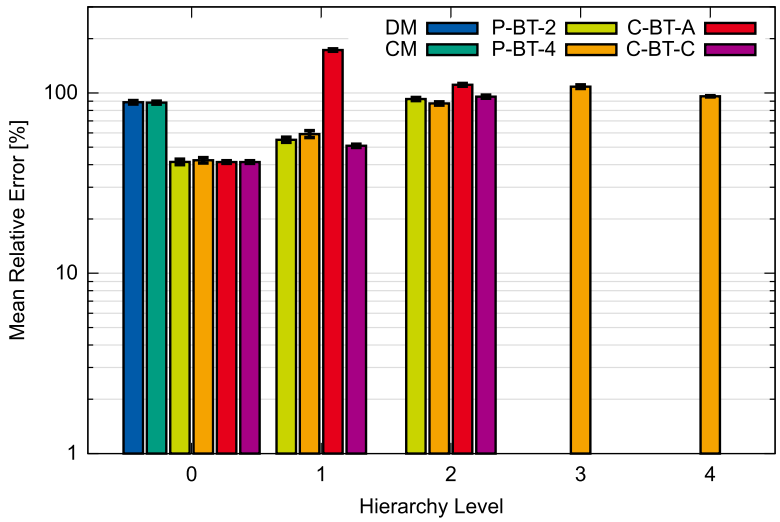


Figure 77: Mean relative error for the node density

update interval for the variation of the noise level attribute over time is configured as listed in Table 41. Based on the different configurations, we examine how the monitoring mechanisms handle these variations. The abbreviation *AVG* indicates that the average over all measurements per virtual square is used. The results for this configuration serve as a reference, because the values of the monitored attribute remain constant during the whole experiment and do not additionally influence accuracy.

Scenario parameter	**Configuration**			
Measurement update interval [s]	10	100	1000	AVG

Table 41: Parameter variation of the measurement update interval. The abbreviation AVG indicates that the average over all measurements per virtual square is used.

6.4.3.2 *Discussion of the Results*

During the discussion of the results, we just focus on accuracy when evaluating the provisioning of location-aware monitoring. The remaining results with respect to timeliness and cost have already been evaluated in the previous sections, where the default scenario has been used.

EVALUATING ACCURACY OF THE MONITORED NODE DENSITY

Starting with the provisioning of location-aware monitoring results regarding the node density per virtual square, Figure 77 depicts the mean relative error of the monitoring mechanisms. In general, the results outline that all approaches exhibit problems to accurately monitor and estimate the node density per virtual square. Even for the node density estimation of the own virtual square, the four hierarchical variants P-BT-2, P-BT-4, C-BT-A, and C-BT-C come to a mean relative error of 40 %. The reason for the overall bad performance of all approaches originates from the properties of the monitored attribute. The overall average node density comes to ≈ 4.21 nodes per virtual square, whereas virtual squares at the border of the modeled area exhibit even a lower density, as depicted in Figure 78. If the current estimate of a

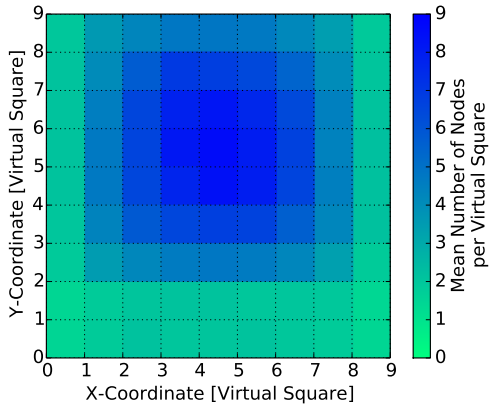


Figure 78: Mean number of nodes per virtual square

virtual square assumes one or two nodes in addition, which just might have left the virtual square, the relative error considerably increases, especially for the sparsely populated virtual squares at the borders of the modeled area.

Apart from this general problem, the results illustrate the provisioning of location-aware monitoring results, as indicated by the lower mean relative error at l_0 and l_1 of P- and C-BlockTree.KOM compared to the two flat variants of Mobi-G.KOM. Due to the hierarchical structure and the provisioning of granular information at lower levels BlockTree.KOM provides a more detailed and accurate view of virtual squares in the vicinity. Particularly, P-BlockTree.KOM benefits from the dissemination of collected information among sibling sectors compared to C-BlockTree.KOM, where data are concentrated at the responsibility blocks of the corresponding sector. The results for C-BT-A indicate that this concentration already leads to a degrading accuracy at l_1. In general, only a fraction of nodes is able to access detailed information of the neighboring blocks and sectors, whereas the majority of nodes must rely on the aggregated results from the concentrating block. The outcome for C-BT-C, which just comprises the monitoring results from nodes inside concentrating blocks, underpins this observation for l_1 and l_2.

If the node density of a virtual square is estimated based on the monitoring data from the hierarchy level that covers the whole MANET, the hierarchical approaches exhibit a similar performance as Mobi-G.KOM CM and DM. With respect to P-BT-4, which operates on a hierarchy with five levels (from l_0 to l_4), the results unveil that the larger hierarchy with smaller sectors per level does not provide any gain in accuracy with respect to this attribute.

To understand the reasons for the discussed outcome we take a look at the distribution of the relative error of all collected measurements, as depicted by the box plot in Figure 79. The distribution of the results of Mobi-G.KOM DM and CM illustrates that the low accuracy arises from an overestimation of the node density. The overestimation particularly originates from the low density at outer virtual squares, which considerably ranges below the calculated average density that Mobi-G.KOM uses as estimate for the node density per virtual square. In contrast, the utilization of the calculated average node density leads to an underestimation in densely populated virtual squares, for instance, in the center of the modeled area, as depicted in Figure 78. However, the illustrated underestimation just partially accounts for the

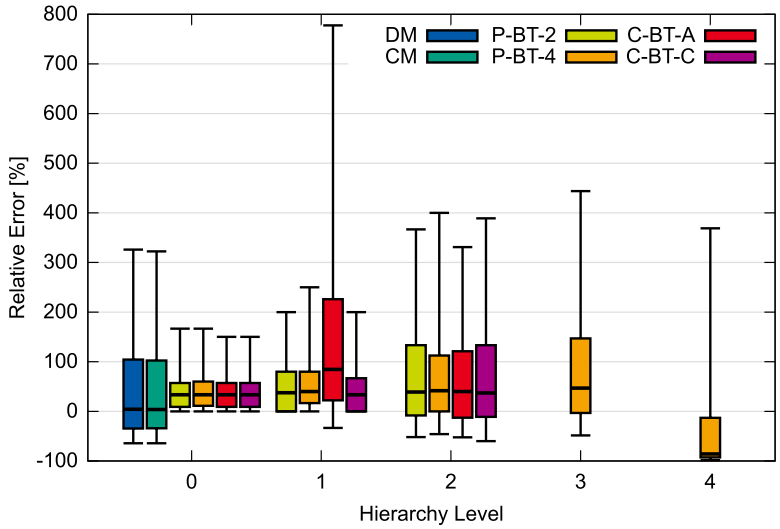
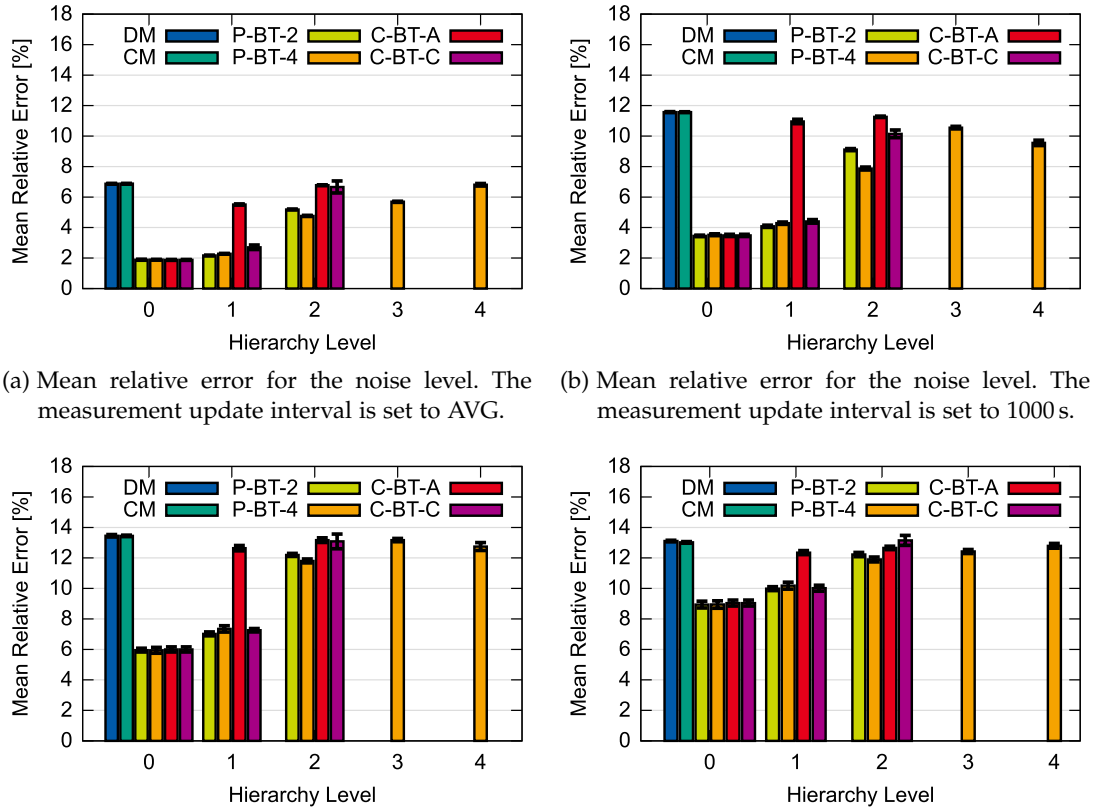


Figure 79: Distribution of the relative error for the node density of all collected measurements

high mean relative error, which is dominated by a considerable overestimation of the current state inside a virtual square (more than 25 % of the results exhibit a mean relative error above 100 %).

Similar tendencies are observable for the hierarchical approaches at l_2 and l_3, where the aggregated data from a large area serves as basis to calculate the node density per virtual square. Similar to Mobi-G.KOM the calculated results considerably overestimate the node density, particularly in outer virtual squares, whereas less than 50 % of the results (except for P-BT-4) underestimate the state. At l_0 and l_1, the presented results reveal that all hierarchical approaches suffer from an overestimation, while an underestimation is not observable for the illustrated 95 % of measurements. This observation is explained by the fact that nodes inside a new block immediately start a Leaf-Broadcasting operation to update the state in that block as well as in the sector at the higher level. Consequently, the information from the new node is immediately reflected in the current state. In contrast, it might happen that the information in a node's previous block and higher sector is not deleted and subsequently reflected in the estimated density per virtual square, which explains the illustrated overestimation.

The results for C-BT-A underpin the discussed negative impact of utilizing the aggregated data from concentrating blocks, which results in a considerable overestimation. The reason for the high outliers (more than 25 % of the results exhibit a relative error above 200 %) stems from virtual squares with a high node density. At l_1, these virtual squares have a stronger impact on the counted number of nodes in the corresponding sector than at l_2, which serves as basis to calculate the density for the remaining virtual squares. As a result the difference between the estimated and the effective node density increases and becomes apparent by the illustrated outliers. Finally, the results of P-BT-4 at l_4 show a completely different behavior than at the lower levels. The phenomenon of the high underestimation is explained by the fact that only a fraction of the fourth level overlaps with the populated area, whereas the larger fraction of the fourth level is not populated. Consequently, the small amount of nodes serves as reference for the calculations at l_4, leading to a low density, which underestimates the effective density in these virtual squares.



(a) Mean relative error for the noise level. The measurement update interval is set to AVG.

(b) Mean relative error for the noise level. The measurement update interval is set to 1000 s.

(c) Mean relative error for the noise level. The measurement update interval is set to 100 s.

(d) Mean relative error for the noise level. The measurement update interval is set to 10 s.

Figure 80: Overview on the mean relative error for the noise level

EVALUATING ACCURACY OF THE MONITORED NOISE LEVEL

The outcome of the provisioning of location-aware monitoring results for the noise level attribute is depicted in Figure 80 and encompasses the plots for the four different configurations of the measurement update interval. The parameter specifies the variation of the attribute over time and is configured as listed in Table 41. Figure 80a depicts the results of MOBI-G.KOM and BLOCKTREE.KOM using a measurement update interval that is set to AVG. This setting indicates that the average per virtual square is used and that the value of a square remains constant over time. We use this configuration as reference to assess the accuracy of the location-aware monitoring results, because the outcome is not influenced by the changing measurements of the attribute but reflects the accuracy of the corresponding approach.

In general, Figure 80a already indicates that the accuracy of monitoring this attribute is considerably higher than of monitoring the node density, which mainly originates from two aspects: instead of the count, we rely on the average as aggregation function, which mitigates the effect of old or missing values; moreover, the dynamics of the underlying MANET do not influence the variability of the monitoring attribute, which is separately configurable by the measurement update interval.

Based on the resulting mean relative error, it is observable that both approaches of BLOCKTREE.KOM provide location-aware monitoring results, whereas MOBI-G.KOM suffers from its flat topology. The design of MOBI-G.KOM does not provide a granular view of nearby regions, while summarizing the state from different regions.

Instead, due to the flat topology, the global view of the attribute serves as reference to estimate the state of nearby and distant virtual squares, which is reflected by the increased mean relative error. In contrast, BlockTree.KOM exploits its hierarchical structure and uses the collected monitoring data from the leaf information tables as well as from the lower levels of the hierarchy or result tables for an accurate prediction of the current state of nearby virtual squares. Especially P-BlockTree.KOM benefits from its hierarchical structure and the dissemination of collected data among sibling sectors. Even at the maximum level l_2, P-BT-2 provides a better estimation of the current situation in distant virtual squares than C-BlockTree.KOM or Mobi-G.KOM. For P-BT-4 the results indicate that the higher hierarchy slightly improves the provided results, at least at l_2. However, the resulting mean relative error at l_3 and l_4 exceeds the one from P-BT-2 at l_2. Though, a smaller area is covered by the federated sectors, the federation of four instead of nine sectors is responsible for the increased relative error of P-BT-4 at l_3 and l_4. With respect to C-BlockTree.KOM the results of C-BT-A and C-BT-C exhibit similar characteristics as for the node density per virtual square and underpin the observation that the concentration of data at single blocks leads to a degrading performance. The depicted results of C-BT-C just comprise the estimates from nodes in concentrating blocks, which access the granular data in the hierarchy table and do not depend on aggregated information from higher levels.

The remaining plots in Figure 80 highlight the influence of the measurement update interval, which increases the frequency of the variation of the attribute over time. Figure 80b and 80c show the results for a measurement update interval of 1000 s and 100 s, respectively. The results exhibit similar tendencies as for the experiment with the static configuration of the noise level attribute. BlockTree.KOM provides an accurate view of neighboring virtual squares, as outlined by the results at l_0 and l_1, whereas the performance at higher levels is comparable to Mobi-G.KOM. Besides, the results reflect that a shorter measurement update interval, which leads to a faster changing attribute, decreases the monitoring accuracy of all mechanisms. However, it is observable that the impact on both flat approaches is stronger than on the hierarchical approaches, because the differences between the results of Mobi-G.KOM at l_0 and P-BT-2, P-BT-4, and C-BT-C at l_0 and l_1 increase. Based on the results for a measurement update interval of 1000 s and 100 s, we draw the conclusion that the advantages of a hierarchical structure become particularly apparent if the attribute changes over time. However, as shown by the last experiment with a measurement update interval of 10 s (cf. Figure 80d) and as already indicated by the outcome for the estimation of node density per virtual square, monitoring a strongly fluctuating attribute reduces the accuracy of a monitoring mechanism. As a result, the differences between flat and hierarchical approaches diminish, as illustrated by the increased mean relative error of the hierarchical approaches at lower levels. The reason for the degrading performance originates from the problems to accurately monitor a highly fluctuating attribute.

6.4.3.3 *Summary on Location-Awareness*

The results from the evaluation of location-aware monitoring underpin the advantages of an underlying hierarchical monitoring topology for the provisioning of location-aware results. In the hierarchical topology the lower levels serve for the

collection and dissemination of local data to provide an accurate and detailed view of a node's vicinity. At higher levels, the hierarchical monitoring mechanisms offer a broader view of the network, which is generated based on the detailed input from the lower levels. Due to the integration and aggregation of data from multiple nodes, which are distributed over a larger area, the accuracy of the calculated state of distant virtual squares degrades. Besides these general observations for BlockTree.KOM, the results for P- and C-BlockTree.KOM indicate how the designed monitoring topology and data collection procedure influence the accuracy of the provided results. A concentration at single parent blocks considerably degrades the accuracy in contrast to the exchange of data between multiple sibling sectors.

Mobi-G.KOM does not operate on a hierarchical but a flat topology, which does not enable the generation of a granular view of a node's vicinity and the provisioning of a summarizing view of broader regions. Instead, the nodes must rely on the global view of a monitored attribute to estimate the state of nearby and distant virtual squares. Consequently, the estimated state does not correctly reflect the state of any square, because the aggregation of monitoring data from all participating nodes in the network conceals the effective state in the respective virtual square.

6.5 CONCLUSIONS

Based on the obtained results and findings, we conclude that BlockTree.KOM and Mobi-G.KOM represent useful solutions that accurately monitor a MANET. Without any central or dedicated entity, BlockTree.KOM and Mobi-G.KOM successfully integrate all participating nodes of a MANET into the monitoring process. On top of these nodes, the two decentralized monitoring mechanisms tackle the first of the five identified research challenges (cf. Chapter 1). Both solutions succeed in collecting data from every node and provide every node with a recent and accurate global view. Furthermore, the results underpin that BlockTree.KOM and Mobi-G.KOM handle the dynamic nature of MANETs and cope with the characteristics of the wireless communication medium, as identified by the second and third research challenge. Despite the limited communication range over an error-prone communication medium paired with mobile nodes that arbitrarily arrive in and leave the MANET, both approaches (i) collect the locally measured data, (ii) generate a global view upon the processed data, and (iii) subsequently disseminate the results.

However, the results and findings from the comparative evaluation in Section 6.4.1 and 6.4.2 outline that BlockTree.KOM and Mobi-G.KOM perform and behave differently if the settings and conditions of a MANET and the surrounding environment are changed. In fact, it becomes apparent that Mobi-G.KOM represents a scalable and robust solution that benefits from its flat topology, which does not require any active maintenance. In combination with the robust communication pattern of gossiping, performance and cost of Mobi-G.KOM are not severely influenced if the node density or the spatial network size is varied. Moreover, Mobi-G.KOM handles even fast moving nodes and copes with stale position information, which is primarily required by the underlying communication methods. Nevertheless, the flat approach exhibits problems in the presence of short-lived nodes, because the time-based restart mechanism, which divides the time into epochs and cycles, does not correctly process data if the nodes leave the MANET. Contrary to Mobi-G.KOM, BlockTree.KOM does

not rely on a restart mechanism but continuously executes its operation without the need for a special treatment of arriving and leaving nodes. Consequently, the variation of the mean node session length has little influence on its performance and cost. Though, BlockTree.KOM has been designed to operate in dynamic networks, the results uncover that the identified concepts for the establishment of the hierarchy and the communication do not suffice to handle increasing network dynamics or networks of a varied spatial size or node density. On the one hand, the results indicate that BlockTree.KOM may benefit from the changing conditions and settings in a MANET. For instance, the underlying hierarchy helps to structure the nodes and to organize the collection and dissemination of nodes so that a large area or a densely populated MANET are timely served with accurate results. On the other hand, both approaches of BlockTree.KOM exhibit considerable problems in sparsely populated networks or in the presence of fast moving nodes. The results of C-BlockTree.KOM additionally uncover that the approach exhibits severe problems in networks of an arbitrary size if the underlying hierarchy is not properly established.

Taking a look at the arising cost with a focus on power consumption, the executed experiments unveil that both approaches of BlockTree.KOM consume considerably more power than Mobi-G.KOM due to the increased traffic. Both hierarchical approaches suffer from the problem that information must be collected and disseminated at multiple levels, which increases the number of messages, every node receives and transmits. In addition, BlockTree.KOM exclusively and excessively relies on contention-based forwarding schemes, which generate a huge amount of traffic in densely populated MANETs. C-BlockTree.KOM minimizes the negative impact of a hierarchy as well as of the utilized contention-based forwarding schemes, because a block just participates at one level. Furthermore, the approach relies on region geocast instead of area dissemination to deliver messages to the addressed block. Contrary to the hierarchical approaches, both variants of Mobi-G.KOM rely on a flat topology with a single level. As a result every node belongs to exactly one level and must not forward data from other levels, which becomes apparent by the considerably decreased traffic. If it is not necessary to monitor fluctuating attributes, Mobi-G.KOM DM may be chosen. The variant minimizes the overall traffic and the resulting power consumption to meet the fourth research challenge and to preserve the scarce resource, as discussed in Chapter 1.

The examination of the provisioning of location-aware monitoring information in Section 6.4.3 reveals the advantages of a hierarchical monitoring topology. While accounting for a large fraction of the overall traffic and the accompanied power consumption, C-BlockTree.KOM and, specifically P-BlockTree.KOM, benefit from the hierarchical monitoring topology, tackling the fifth research challenge (cf. Chapter 1). At lower levels, P- and C-BlockTree.KOM provide an accurate and detailed view of a node's vicinity. At higher levels, both approaches offer a broader view of the network but with a decreasing accuracy due to the integration of data from multiple nodes. With its one-tiered topology, Mobi-G.KOM operates at considerably lower cost, but the results unveil that the solution cannot provide location-aware monitoring information. To estimate the current situation of nearby as well as distant regions, Mobi-G.KOM must rely on the global view of a monitored attribute, which leads to the observed inaccurate estimation of both nearby and distant locations.

Table 42 and 43 summarize and the previously discussed findings of the comparative evaluation for BlockTree.KOM and Mobi-G.KOM.

Scenario parameter	BlockTree.KOM	Mobi-G.KOM
Spatial network size	• Medium influence on performance • C-BlockTree.KOM cannot handle networks of an arbitrary spatial size. • Medium influence on C-BlockTree.KOM's cost • Strong influence on P-BlockTree.KOM's cost	• Little influence on performance and cost • Degrading performance in networks of a very large spatial size
Node density	• Strong influence on C-BlockTree.KOM's performance • Medium influence on P-BlockTree.KOM's performance • Both approaches, specifically C-BlockTree.KOM, exhibit a degrading performance in sparsely populated networks. • Strong influence on cost	• Little influence on performance • Medium influence on cost
Provisioning of location-aware monitoring information	• P-BlockTree.KOM provides location-aware monitoring information at every level of the hierarchy. • In C-BlockTree.KOM only a fraction of nodes is provided with location-aware monitoring information at every level of the hierarchy. The majority of nodes only obtains location-aware monitoring information at the lowest level of the hierarchy.	• No provisioning of location-aware monitoring information

Table 42: Summary of the outcome and the influence of the varied scenario parameters on BlockTree.KOM and Mobi-G.KOM. The related experiments consist of the experiments for scalability as well as for the examination of the provisioning of location-aware monitoring information.

Scenario parameter	BlockTree.KOM	Mobi-G.KOM
Movement speed	• Strong influence on performance • Little influence on cost • BlockTree.KOM cannot operate in the presence of fast moving nodes as reflected by the decreased accuracy due to a reduced data exchange.	• Little influence on performance and cost
Churn	• Little influence on performance and cost	• Medium influence on performance • Little influence on cost
GPS utilization	• Little influence on performance • BlockTree.KOM exhibits a degrading performance for a very long position refresh interval. • Little influence on C-BlockTree.KOM's cost • Medium influence on P-BlockTree.KOM's cost	• Little influence on performance • Little influence on Mobi-G.KOM DM's cost • Medium influence on Mobi-G.KOM CM's cost

Table 43: Summary of the outcome and the influence of the varied scenario parameters on BlockTree.KOM and Mobi-G.KOM. The related experiments consist of the experiments for robustness.

7 CONCLUSION

During the final chapter, we conclude our thesis and elaborate on the future work in the area of decentralized monitoring in MANETs. In Section 7.1, we summarize the content of the previous six chapters, present the main contributions and draw conclusions based on the obtained results. Subsequently, Section 7.2 provides an outlook on open issues and future work. With Section 7.3, we conclude the thesis with some final remarks.

7.1 SUMMARY OF THE THESIS

At the beginning of this thesis, we introduced MANETs and argued on their usefulness as communication substrate that complements or substitutes an existing cellular infrastructure. Thereupon, we motivated the need for monitoring to provide a recent view of the state of this class of communication networks, serving as input for transitions or multi-mechanisms adaptations of the communication network. Based on the characteristics of decentralized monitoring and MANETs, we (i) identified arising research challenges, (ii) argued about shortcomings of existing approaches, and, consequently, (iii) formulated the *main objective* of our thesis: "the design of decentralized monitoring approaches in MANETs to provide accurate and location-aware monitoring information about the current network state". To ease the understanding of our contributions and design decisions we provided the technical background on related research areas in Chapter 2. First, we introduced the envisaged application scenarios to sketch the arising requirements and demands, which influence the MANET as well as the deployed decentralized monitoring mechanism. Afterwards, we outlined common concepts of communication networks with an outlook on transition-enabled networks, as envisaged by the Future Internet project MAKI. The outlook already emphasized the need for monitoring in transition-enabled communication networks, because it provides the required insights into the current state of the network and its participating nodes that serve as basis for transitions and multi-mechanisms adaptations. Subsequently, we outlined the characteristics and properties of MANETs, comprising (i) the nomenclature and definitions for routing in MANETs as well as (ii) an overview on protocols to route information through the network. To conclude Chapter 2 we introduced the topic of monitoring in communication networks and identified the corresponding class of network management our envisaged monitoring mechanisms belong to. On this basis, we presented the principles of decentralized monitoring, covering the determination of the three main components of a decentralized monitoring mechanism as well as the identification of its functional and non-functional requirements.

7.1.1 *Contributions*

Given the basics, we performed an extensive survey in Chapter 3, constituting the *first contribution* of the thesis and comprising a detailed analysis of decentralized monitoring mechanisms. For the analysis we surveyed existing approaches for fixed communication networks in addition to approaches for MANETs to broaden our overview. We examined how existing solutions implement the three major components of a decentralized monitoring mechanism and identified further categories per component. In addition to the three main monitoring components we separately examined different techniques for data aggregation, which represents an important aspect of data collection. Based on the three main components and the identified categories per component, we classified existing approaches for decentralized monitoring in fixed and mobile communication networks. The performed classification has been summarized in Figure 9 and 10, constituting a landscape for decentralized monitoring mechanisms in fixed and mobile communication networks.

After elaborating the background of our decentralized monitoring mechanisms, we introduced BlockTree.KOM in Chapter 4, representing our *second contribution* of the thesis as well as our first decentralized monitoring mechanism for MANETs. With BlockTree.KOM we decided to develop an approach that operates on a hierarchical monitoring topology, which is used for the collection of data as well as dissemination of results. The decision for a hierarchical approach despite the deployment in a highly dynamic environment stems from the fact that a hierarchy eases the provisioning of location-aware monitoring information. To cope with the dynamic nature of MANETs and to handle the characteristics of the wireless communication medium we identified four major concepts to maintain a hierarchy and to exchange information over that hierarchy. As a consequence, BlockTree.KOM operates on a *tree of clusters*, more precisely *tree of blocks*. To avoid the dependency of BlockTree.KOM on a monitoring-agnostic routing protocol it uses a dedicated set of simple yet robust communication methods to exchange information even if the underlying routing mechanism fails, as demanded by Nanda and Kotz [119]. The designed communication methods are based on concepts of contention-based forwarding schemes and have been tailored to BlockTree.KOM's demands. Contrary to other approaches, which either provide the generated results to third parties [119, 142], such as network operators, or where the results are located at a set of nodes [137, 97, 8, 142], BlockTree.KOM distributes the required results among the remaining nodes in the network so that every node has a recent view of the current state of the network.

With C-BlockTree.KOM and P-BlockTree.KOM we developed two concrete approaches that rely on the presented communication methods and on the four concepts of BlockTree.KOM but implement the presented concepts in different ways.

1. In P-BlockTree.KOM a block participates at every level of the hierarchy and actively triggers the corresponding operations for every level. For the creation of the underlying tree sectors from the same level are federated and form a sector at the next higher level. Among the federated sectors, the monitored data are collected and implicitly disseminated so that a separate phase with dedicated operations for the dissemination of monitoring results is not necessary.

2. In C-BlockTree.KOM blocks just belong to one level. The exchange of information does not take place between sibling sectors but between parent and child

blocks. Contrary to P-BlockTree.KOM, higher levels of the hierarchy do not consist of federations of sectors but are represented by single blocks. At these blocks, the monitoring data are concentrated, processed, and forwarded to the next higher level, requiring a separate phase and operation to disseminate the monitoring results.

Mobi-G.KOM constitutes our *third contribution* of the thesis, representing the second approach for decentralized monitoring in MANETs. As described in Chapter 5 we decided to develop a solution that significantly differs from BlockTree.KOM to investigate the influence of its differing design on performance and cost. Furthermore, another aim has been to promote the development of decentralized monitoring mechanisms for MANETs that operate on a completely flat topology. Currently, only a small number of solutions rely on a flat topology to monitor a MANET [180, 41, 50], as discussed in Chapter 3. Consequently, Mobi-G.KOM operates on top of a flat monitoring topology and relies on the robust communication pattern of gossiping for the communication between the nodes. Contrary to the majority of flat and gossip-based approaches [69, 170, 139], Mobi-G.KOM does not apply concepts of mass conservation, where gossiping is used to aggregate data. Instead, Mobi-G.KOM gossips to exchange information, using concepts from population algorithms [5] to process and aggregate the data. On the one hand, due to the selection of a flat monitoring topology, we sacrifice the possibility to provide location-aware monitoring results. On the other hand, Mobi-G.KOM (i) must not maintain a hierarchy and (ii) exploits the robust properties of gossiping to exchange information. We developed two variants of Mobi-G.KOM that enable discrete and continuous monitoring. For the communication we adopted routing methods from BlockTree.KOM so that Mobi-G.KOM operates independently from an underlying, monitoring-agnostic routing protocol.

7.1.2 *Conclusions*

Chapter 6 dealt with the evaluation of both approaches of BlockTree.KOM and both variants of Mobi-G.KOM. The goal of the evaluation addressed the major objective of thesis and quantified to which extent both approaches provide accurate and location-aware monitoring information in MANETs. To answer that question an extensive set of experiments has been conducted, consisting of two separate parts. In the first part we performed a system parameter evaluation (i) to understand at which cost the approaches perform and (ii) to examine the sensitivity and possible tradeoffs of varied system parameters. In the second part we compared BlockTree.KOM and Mobi-G.KOM with each other. The corresponding scenarios of the experiments have been derived from the workload-dependent non-functional requirements of decentralized monitoring mechanisms with a focus on scalability and robustness. For the quantification of performance and cost we derived a set of metrics from the workload-independent non-functional requirements.

The obtained results outline that the two approaches of BlockTree.KOM as well as the two variants of Mobi-G.KOM tackle the first of our identified research challenges (cf. Section 1.2): all approaches and variants successfully collect and process the monitoring data, followed by a timely dissemination of the accurate monitoring results among the participating nodes. As demanded by the second and third research challenge BlockTree.KOM and Mobi-G.KOM operate in an inherently dy-

namic network relying on a wireless and error-prone communication medium with a limited communication range. The findings of the system parameter evaluation already reveal that Mobi-G.KOM benefits from its flat topology and the gossip-based information exchange: it accurately monitors the specified attributes at low cost. Contrary, both approaches of BlockTree.KOM generate more traffic and consume more power than both variants of Mobi-G.KOM to achieve a comparable accuracy. In this context the obtained results underpin that Mobi-G.KOM meets the fourth challenge and reduces the power consumption to a minimum, whereas C- and, specifically, P-BlockTree.KOM dissipate the scarce resource of energy.

The experiments of the comparative evaluation especially focused on the second research challenge and examined the influence of (i) node mobility, (ii) arbitrary node arrival and departure, also referred to as churn (cf. Section 2.1.1), (iii) networks of different spatial sizes, and (iv) networks of different densities. During the comparative evaluation, significant differences between the flat and hierarchical approaches were observable. In terms of scalability, Mobi-G.KOM benefits from its design, since neither the underlying topology nor the information exchange are severely influenced by a varied node density or spatial network size. Consequently, both variants of Mobi-G.KOM provide accurate results, while the traffic slightly increases as a function of the node density or the spatial network size. BlockTree.KOM exhibits considerable problems in sparsely populated scenarios but defeats Mobi-G.KOM in networks of a very large spatial size or with a very high node density. The two approaches profit from their hierarchical topology, which structures the nodes and organizes data collection and result dissemination so that large groups of nodes as well as distant nodes are timely provided with accurate results. However, C-BlockTree.KOM cannot operate in networks of an arbitrary size, because it requires a fully established hierarchy for its proper operation.

With respect to robustness, the results have revealed that Mobi-G.KOM handles even fast moving nodes and copes with stale position information. In this regard Mobi-G.KOM DM and CM must not establish and manage a hierarchy on top of fast moving nodes or with inaccurate position information. Furthermore, the robust, gossip-based communication ensures that the collected monitoring data as well as the disseminated monitoring results reach their destinations. However, Mobi-G.KOM exhibits problems with short-lived nodes, because the time-based organization does not purge measured data from short-lived nodes, which is reflected by the decreased accuracy.

The experiments that examined the provisioning of location-aware monitoring results revealed that Mobi-G.KOM cannot provide location-aware monitoring results and, thus, does not meet our fifth research challenge. Whereas the flat topology (i) helps to handle fast moving nodes or inaccurate position information and (ii) mitigates the influence of large spatial or densely populated networks on the arising cost, it sacrifices the possibility to provide location-aware monitoring results. In this regard particularly P-BlockTree.KOM benefits from its hierarchical monitoring topology: at lower levels, nodes obtain a detailed view of their vicinity, whereas the higher levels offer information for larger areas but of a reduced granularity.

To conclude, BlockTree.KOM and Mobi-G.KOM represent two valuable solutions for decentralized monitoring in MANETs. They accurately monitor a set of specified attributes and timely distribute the generated results, despite the dynamic nature of MANETs paired with the characteristics of the wireless communication medium.

With MOBI-G.KOM DM and CM we developed a flat monitoring mechanism, which avoids the costly creation and maintenance of a hierarchy but exchanges information at a single level. Both variants exploit the robust communication pattern of gossiping, but solely apply this pattern for communication instead of aggregation. Consequently, MOBI-G.KOM represents a highly robust decentralized monitoring mechanism that monitors MANETs at very low cost, preserving the limited amount of disposable energy. Contrary to MOBI-G.KOM, the two approaches of BLOCKTREE.KOM operate on a hierarchical topology, more precisely on trees of blocks. Based on the four identified concepts, P- and C-BLOCKTREE.KOM withstand the inherently dynamic nature of MANETs and the characteristics of the wireless communication medium. Despite the sophisticated and robust topology design, the results unveil that both approaches are very susceptible to varying conditions in the network, which becomes apparent by the varying performance or cost. However, the maintenance of the hierarchy accompanied by the increased traffic and power consumption pays off, if the provisioning of location-aware monitoring results is required. Thanks to multiple hierarchy levels, which manage data of different granularity for areas of different sizes, C- and, specifically, P-BLOCKTREE.KOM provide location-aware monitoring results.

7.2 OUTLOOK

Based on our contributions in this thesis, we have taken a further step towards the provisioning of accurate, location-aware monitoring information in MANETs. The developed decentralized monitoring mechanisms BLOCKTREE.KOM and MOBI-G.KOM accurately capture the current state in the MANET and provide each participating node with this information. As emphasized during the introduction of the thesis (cf. Chapter 1) and during the presentation of transition-enabled communication networks (cf. Section 2.2), these valuable insights may serve as basis for a transition or multi-mechanisms adaptation of the communication network. A transition or multi-mechanisms adaptation may also affect and incorporate the decentralized monitoring mechanism itself. Consequently, decentralized monitoring mechanisms must also be able to execute a single transition or a transition as part of a multi-mechanisms adaptation so that the network is monitored before, during, and after the transition. During the remainder of this section, we elaborate on future decentralized monitoring mechanisms, covering transition-enabled mechanisms and mechanisms for other application areas that go beyond MANETs.

Future decentralized monitoring mechanisms may use the provided monitoring results to adapt their current configuration. As shown by our results the traffic of P- and C-BLOCKTREE.KOM increases with a higher node density in the network, because the redundant contention-based forwarding schemes lead to an increased amount of concurrent transmissions. BLOCKTREE.KOM might use the information about the current density to reduce the sets of sources $\mathcal{S}$ and intermediate nodes $\mathcal{I}$ by applying of a random-based threshold, which depends on the node density. If nodes additionally track their movement speed, this information may be exploited to adjust the periodic execution of operations. In terms of BLOCKTREE.KOM, the aggregation interval may be shortened or prolonged, while the token-send-delay and/or the refresh timeout of MOBI-G.KOM might be adjusted.

Besides adapting the configuration of a single decentralized monitoring mechanism, it is worth investigating methods that facilitate the transition of a multi-mechanism for monitoring between functionally equivalent monitoring mechanisms. Depending on the current state of the communication network, a currently deployed decentralized monitoring mechanism might be replaced by another one that exhibits a better performance or reduces the cost at the given conditions. Considering, for example, a varying node density in a MANET, Mobi-G.KOM might serve as fallback if the node density drops below a certain threshold so that BlockTree.KOM cannot operate correctly. In contrast, if the mean node session time decreases, BlockTree.KOM constitutes an option to replace Mobi-G.KOM.

So far, the thesis exclusively dealt with MANETs, but, as indicated in Chapter 1 and as surveyed by Cavalcanti et al. [20], MANETs are also considered as a complement to cellular networks. Consequently, prospective decentralized monitoring mechanisms might be deployed on top of these heterogeneous networks, spanning over the mobile as well as the cellular network to collect information from nodes in both networks. Here, it is also worth investigating how and where the collected monitoring data may be disseminated and which nodes may be provided with the monitoring results.

7.3 FINAL REMARKS

The vast distribution and increased utilization of mobile communication devices enables and partially urges the deployment of mobile ad hoc networks. Especially in urban areas, which are densely populated with users that carry a mobile communication device, MANETs serve as an useful and viable communication substrate. This substrate either complements cellular networks to exchange information or serves as an alternative if cellular networks are not available. To maintain a MANET and react on changes that arise from the dynamic nature of the communication network or originate from the varying conditions of the surrounding environment, the adaptation of those networks is inevitable. Given the current state of the MANET, the incorporated communication mechanisms of the network may be adapted so that the communication network conforms to the identified state. In this context, monitoring constitutes a mandatory element to determine and provide the required insights into the current network state and the participating nodes. Based on a set of specified attributes that characterize the current state of the network and the participating nodes, a monitoring mechanism collects and processes the set of attributes and disseminates the insights about the current state. Accordingly, the thesis deals with the objective *to provide accurate and location-aware monitoring information*. With BlockTree.KOM and Mobi-G.KOM we developed two decentralized monitoring mechanisms that address this objective in MANETs with the identified research challenges in mind. BlockTree.KOM enables the provisioning of location-aware monitoring results due to its hierarchical monitoring topology. With Mobi-G.KOM we sacrifice this functionality but obtain a highly robust decentralized monitoring mechanism, operating at very low cost.

7.4 ACKNOWLEDGMENTS

This work has been co-funded by the German Research Foundation (DFG) within the Researcher Group 733 "QuaP2P – Improvement of the Quality of Peer-to-Peer Systems by Systematically Researching Quality Features and their Interdependencies", as part of Project B01 within the Collaborative Research Center 1053 "MAKI – Multi-Mechanisms Adaptation for the Future Internet", and by the Social Link Project within the Loewe Program of Excellence in Research, Hessen, Germany.

BIBLIOGRAPHY

[1] Osama Abboud. *Quality Adaptation In Peer-to-Peer Video Streaming: Supporting Heterogeneity and Enhancing Performance using Scalable Video Coding*. PhD thesis, Technische Universität Darmstadt, 2012.

[2] Keno Albrecht, Ruedi Arnold, Michael Gahwiler, and Roger Wattenhofer. Aggregating Information in Peer-to-Peer Systems for Improved Join and Leave. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, 2004.

[3] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A Survey of Peer-to-Peer Content Distribution Technologies. *ACM Computing Surveys*, 36(4):335–371, 2004.

[4] Nils Aschenbruck, Raphael Ernst, Elmar Gerhards-Padilla, and Matthias Schwamborn. BonnMotion: a Mobility Scenario Generation and Analysis Tool. In *Proceedings of the International ICST Conference on Simulation Tools and Techniques*, 2010.

[5] James Aspnes and Eric Ruppert. An Introduction to Population Protocols. *Bulletin of the European Association for Theoretical Computer Science*, 93:98–117, 2007.

[6] Xuan Bao, Yin Lin, Uichin Lee, Ivica Rimac, and Romit R. Choudhury. DataSpotting: Exploiting Naturally Clustered Mobile Devices to Offload Cellular Traffic. In *Proceedings of the 32nd IEEE International Conference on Computer Communications*, 2013.

[7] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the 4th Annual ACM/IEEE International Conference on Mobile Computing and Networking*, 1998.

[8] Nadia Battat and Hamamache Kheddouci. HMAN: Hierarchical Monitoring for Ad Hoc Networks. In *Proceedings of the IFIP International Conference on Embedded and Ubiquitous Computing*, 2011.

[9] Mayank Bawa, Aristides Gionis, Hector Garcia-Molina, and Rajeev Motwani. The Price of Validity in Dynamic Networks. *Journal of Computer and System Sciences*, 73(3):245–264, 2007.

[10] Paolo Bellavista, Axel Küpper, and Sumi Helal. Location-Based Services: Back to the Future. *IEEE Pervasive Computing*, 7(2):85–89, 2008.

[11] Paolo Bellavista, Antonio Corradi, and Luca Foschini. Self-Organizing Seamless Multimedia Streaming in Dense Manets. *IEEE Pervasive Computing*, 12(1): 68–78, 2013.

[12] Jeff Boleng, Tracy Camp, and Vishy Tolety. Mesh-Based Geocast Routing Protocols in an Ad Hoc Network. In *Proceedings of the 15th International Parallel & Distributed Processing Symposium*, 2001.

[13] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with Guaranteed Delivery in Ad Hoc Wireless Networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, 1999.

[14] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized Gossip Algorithms. *IEEE Transactions on Information Theory*, 52(6):2508–2530, 2006.

[15] Robert Braden. Requirements for Internet Hosts – Application and Support. RFC 1123, RFC Editor, 1989. URL `http://www.ietf.org/rfc/rfc1123.txt`.

[16] Ilja N. Bronstein, Konstantin A. Semendjajew, Gerhard Musiol, and Heiner Mühlig. *Taschenbuch der Mathematik*. Harri Deutsch, 2008.

[17] Canalys. Smart Phones Overtake Client PCs in 2011, 2012. URL `http://www.canalys.com/static/press_release/2012/canalys-press-release-030212-smart-phones-overtake-client-pcs-2011_0.pdf`. Last accessed on July 18, 2014.

[18] Justin Cappos and John H. Hartman. San Fermín: Aggregating Large Data Sets Using a Binomial Swap Forest. In *Proceedings of the 5th Symposium on Networked Systems Design and Implementation*, 2008.

[19] Alfredo A. V. Castro, Giovanna Di Marzo Serugendo, and Dimitri Konstantas. Hovering Information – Self-Organizing Information that Finds its Own Storage. In Athanasios V. Vasilakos, Manish Parashar, Stamatis Karnouskos, and Witold Pedrycz, editors, *Autonomic Communication*, pages 111–145. Springer, 2009.

[20] Dave Cavalcanti, Dharma P. Agrawal, Carlos de M. Cordeiro, Bin Xie, and Anup Kumar. Issues in Integrating Cellular Networks, WLANs, and MANETs: a Futuristic Heterogeneous Wireless Network. *IEEE Wireless Communications*, 12(3):30–41, 2005.

[21] Wenli Chen, Nitin Jain, and Suresh Singh. ANMP: Ad Hoc Network Management Protocol. *IEEE Journal on Selected Areas in Communications*, 17(8):1506–1531, 1999.

[22] Zhiyuan Cheng, James Caverlee, Kyumin Lee, and Daniel Sui. Exploring Millions of Footprints in Location Sharing Services. In *International AAAI Conference on Weblogs and Social Media*, 2011.

[23] Ching-Chuan Chiang, Hsiao-Kuang Wu, Winston Liu, and Mario Gerla. Routing in Clustered Multihop Mobile Wireless Networks with Fading Channel. In *Proceedings of IEEE Singapore International Conference on Networks*, 1997.

[24] Imrich Chlamtac, Marco Conti, and Jennifer J.-N. Liu. Mobile Ad Hoc Networking: Imperatives and Challenges. *Ad Hoc Networks*, 1(1):13–64, 2003.

[25] Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2013–2018, 2014. URL http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf. Last accessed on July 18, 2014.

[26] Bram Cohen. Incentives Build Robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer Systems*, 2003.

[27] Edith Cohen and Scott Shenker. Replication Strategies in Unstructured Peer-to-Peer Networks. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2002.

[28] Jeffrey Considine, Feifei Li, George Kollios, and John Byers. Approximate Aggregation Techniques for Sensor Databases. In *Proceedings of the 20th International Conference on Data Engineering*, 2004.

[29] Mads Dam and Rolf Stadler. A Generic Protocol for Network State Aggregation. In *Radiovetenskap och Kommunikation RVK*, 2005.

[30] Vasilios Darlagiannis. *Overlay Network Mechanisms for Peer-to-Peer Systems*. PhD thesis, Technische Universität Darmstadt, 2006.

[31] John D. Day and Hubert Zimmermann. The OSI Reference Model. *Proceedings of the IEEE*, 71(12):1334–1340, 1983.

[32] Subhankar Dhar and Upkar Varshney. Challenges and Business Models for Mobile Location-Based Services and Advertising. *Communications of the ACM*, 54(5):121–128, 2011.

[33] Oxford Dictionaries. Defition of "to monitor", 2013. URL http://oxforddictionaries.com/definition/american_english/monitor. Last accessed on July 18, 2014.

[34] Mark Dilman and Danny Raz. Efficient Reactive Monitoring. *IEEE Journal on Selected Areas in Communications*, 20(4):668–676, 2002.

[35] Richard Draves, Jitendra Padhye, and Brian Zill. Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks. In *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, 2004.

[36] Ericsson. Ericsson Mobility Report - On the Pulse of the Networked Society, 2013. URL http://www.ericsson.com/res/docs/2013/ericsson-mobility-report-november-2013.pdf. Last accessed on July 18, 2014.

[37] Patrick T. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, and Laurent Massoulié. Epidemic Information Dissemination in Distributed Systems. *IEEE Computer*, 37(5):60 – 67, 2004.

[38] Laura M. Feeney and Martin Nilsson. Investigating the Energy Consumption of a Wireless Network Interface in an Ad Hoc Networking Environment. In *Proceedings of the IEEE Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, 2001.

[39] Gregory G. Finn. Routing and Addressing Problems in Large Metropolitan-Scale Internetworks. *ISI Resource Report ISU/RR*, pages 87–180, 1987.

[40] Holger Füßler, Jörg Widmer, Michael Käsemann, Martin Mauve, and Hannes Hartenstein. Contention-Based Forwarding for Mobile Ad Hoc Networks. *Ad Hoc Networks*, 1(4):351–369, 2003.

[41] Joanna Geibig and Dirk Bradler. Self-Organized Aggregation in Irregular Wireless Networks. In *IFIP Wireless Days*, 2010.

[42] Ivan A. Getting. Perspective/Navigation - The Global Positioning System. *IEEE Spectrum*, 30(12):36–38, 1993.

[43] Silvia Giordano and Ivan Stojmenović. Position Based Routing Algorithms for Ad Hoc Networks: a Taxonomy. In Xiuzhen Cheng, Xiao Huang, and Ding-Zhu Du, editors, *Ad Hoc Wireless Networking*, pages 103–136. Kluver, 2003.

[44] Christian Gottron. *Security in Mobile Peer-to-Peer Architectures – Introducing Mechanisms to Increase the Robustness of Overlay Routing Algorithms of Mobile Peer-to-Peer Architectures*. PhD thesis, Technische Universität Darmstadt, 2013.

[45] Christian Gottron, André König, and Ralf Steinmetz. A Cluster-Based Locality-Aware Mobile Peer-to-Peer Architecture. In *Proceedings of the 8th International Workshop on Mobile Peer-to-Peer Computing*, 2012.

[46] Kalman Graffi. *Monitoring and Management of Peer-to-Peer Systems*. PhD thesis, Technische Universtiät Darmstadt, 2010.

[47] Kalman Graffi, Dominik Stingl, Julius Rückert, Aleksandra Kovacevic, and Ralf Steinmetz. Monitoring and Management of Structured Peer-to-Peer Systems. In *Proceedings of the 9th International Conference on Peer-to-Peer Computing*, 2009.

[48] Christian Gross, Fabian Kaup, Dominik Stingl, Björn Richerzhagen, David Hausheer, and Ralf Steinmetz. EnerSim: an Energy Consumption Model for Large-Scale Overlay Simulators. In *Proceedings of the 38th IEEE Conference on Local Computer Networks*, 2013.

[49] Christian Gross, Dominik Stingl, Wolfgang Effelsberg, and Ralf Steinmetz. Neuer DFG-Sonderforschungsbereich an der Technischen Universität Darmstadt. *Praxis der Informationsverarbeitung und Kommunikation*, 36(1):65–66, 2013.

[50] Alessio Guerrieri, Iacopo Carreras, Francesco De Pellegrini, Daniele Miorandi, and Alberto Montresor. Distributed Estimation of Global Parameters in Delay-Tolerant Networks. *Computer Communications*, 33(13):1472–1482, 2010.

[51] Indranil Gupta, Robbert van Renesse, and Kenneth P. Birman. Scalable Fault-Tolerant Aggregation in Large Process Groups. In *Proceedings of the IEEE International Conference on Dependable Systems and Networks*, 2001.

[52] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. RFC 2026: the Zone Routing Protocol (ZRP) for Ad Hoc Networks, 2002. URL `http://people.ece.cornell.edu/haas/wnl/Publications/draft-ietf-manet-zone-zrp-04.txt`. Last accessed on July 18, 2014.

[53] Marc Heissenbüttel, Torsten Braun, Thomas Bernoulli, and Markus Wälchli. BLR: Beacon-Less Routing Algorithm for Mobile Ad-Hoc Networks. *Computer Communications*, 27(11):1076–1086, 2004.

[54] Olafur R. Helgason, Emre A. Yavuz, Sylvia T. Kouyoumdjieva, Ljubica Pajevic, and Gunnar Karlsson. A Mobile Peer-to-Peer System for Opportunistic Content-Centric Networking. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Networking, Systems, and Applications on Mobile Handhelds*, 2010.

[55] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. ns-3 Project Goals. In *Proceeding From the Workshop on ns-2: the IP Network Simulator*, 2006.

[56] Matthias Hollick. *Dependable Routing for Cellular and Ad hoc Networks*. PhD thesis, Technische Universität Darmstadt, 2004.

[57] Ting-Chao Hou and Victor O.K. Li. Transmission Range Control in Multihop Packet Radio Networks. *IEEE Transactions on Communications*, 34(1):38–44, 1986.

[58] Hung-Chang Hsiao and Chung-Ta King. Bristle: a Mobile Structured Peer-To-Peer Architecture. In *17th International Symposium on Parallel and Distributed Processing*, 2003.

[59] Christian Hübsch, Christoph P. Mayer, Sebastian Mies, Roland Bless, Oliver P. Waldhorst, and Martina Zitterbart. Reconnecting the Internet with Ariba: Self-Organizing Provisioning of End-To-End Connectivity in Heterogeneous Networks. *ACM SIGCOMM Computer Communication Review*, 40(1):131–132, 2010.

[60] IEEE. IEEE Standard for Information Technology – Telecommunications and Information Exchange Between Systems – Local and Metropolitan Area Networks – Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, pages 1–2793, 2012.

[61] Chalermek Intanagonwiwat, Deborah Estrin, Ramesh Govindan, and John Heidemann. Impact of Network Density on Data Aggregation in Wireless Sensor Networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems*, 2002.

[62] 7498-4 ISO/IEC. Information Processing Systems - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework, 1989. URL `http://standards.iso.org/ittf/PubliclyAvailableStandards/s014258_ISO_IEC_7498-4_1989%28E%29.zip`. Last accessed on July 18, 2014.

[63] Nafaâ Jabeur, Sherali Zeadally, and Biju Sayed. Mobile Social Networking Applications. *Communications of the ACM*, 56(3):71–79, 2013.

[64] Philippe Jacquet, Paul Mühlethaler, Thomas Clausen, Anis Laouiti, Amir Qayyum, and Laurent Viennot. Optimized Link State Routing Protocol for Ad Hoc Networks. In *IEEE International Multi Topic Conference*, 2001.

[65] Navendu Jain, Dmitry Kit, Prince Mahajan, Praveen Yalagandula, Mike Dahlin, and Yin Zhang. PRISM: PRecision Integrated Scalable Monitoring. Technical report, Department of Computer Sciences, University of Texas at Austin, 2006.

[66] Rahul Jain, Anuj Puri, and Raja Sengupta. Geographical Routing Using Partial Information for Wireless Ad Hoc Networks. *IEEE Personal Communications*, 8 (1):48–57, 2001.

[67] Raj Jain. *The Art Of Computer Systems Performance Analysis:*. Wiley India Pvt. Limited, 2008.

[68] Márk Jelasity, Rachid Guerraoui, Anne-Marie Kermarrec, and Maarten van Steen. The Peer Sampling Service: Experimental Evaluation of Unstructured Gossip-Based Implementations. In *Proceedings of the 5th International Conference on Middleware*, 2004.

[69] Márk Jelasity, Alberto Montresor, and Ozalp Babaoglu. Gossip-Based Aggregation in Large Dynamic Networks. *ACM Transactions on Computer Systems*, 23 (3):219–252, 2005.

[70] Xia Jiang and Tracy Camp. A Review of Geocasting Protocols for a Mobile Ad Hoc Network. In *Proceedings of the Grace Hopper Celebration*, 2002.

[71] David B. Johnson and David A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Tomasz Imielinski and Henry F. Korth, editors, *Mobile Computing*, pages 153–181. Springer, 1996.

[72] Jeffrey Joyce, Greg Lomow, Konrad Slind, and Brian Unger. Monitoring Distributed Systems. *ACM Transactions on Computer Systems*, 5(2):121–150, 1987.

[73] Frederik Jungermann. Growing Pains for Crowded Wireless Networks, 2009. URL `http://www.nokiasiemensnetworks.com/news-events/publications/unite-trends-and-insights-2009/trend-two-driving-efficiency/growing-pains-f`. Last accessed on July 18, 2014.

[74] Iris A. Junglas and Richard T. Watson. Location-Based Services. *Communications of the ACM*, 51(3):65–69, 2008.

[75] Dan Jurca and Rolf Stadler. H-GAP: Estimating Histograms of Local Variables with Accuracy Objectives for Distributed Real-Time Monitoring. *IEEE Transactions on Network and Service Management*, 7(2):83–95, 2010.

[76] Charles F. F. Karney. Transverse Mercator with an Accuracy of a few Nanometers. *Journal of Geodesy*, 85(8):475–485, 2011.

[77] Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for Wireless Networks. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, 2000.

[78] Sebastian Kaune. *Performance and Availability in Peer-to-Peer Content Distribution Systems: a Case for a Multilateral Incentive Approach*. PhD thesis, Technische Universität Darmstadt, 2011.

[79] Hanif Kazemi, George Hadjichristofi, and Luiz A. DaSilva. MMAN - A Monitor for Mobile Ad hoc Networks: Design, Implementation, and Experimental Evaluation. In *Third ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, 2008.

[80] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. MicroCast: Cooperative Video Streaming on Smartphones. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, 2012.

[81] David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-Based Computation of Aggregate Information. In *Proceedings of the 44th Annual Symposium on Foundations of Computer Science*, 2003.

[82] Srinivasan Keshav. Efficient and Decentralized Computation of Approximate Global State. *ACM SIGCOMM Computer Communication Review*, 36(1):69–74, 2006.

[83] Wolfgang Kieß, Holger Füßler, Jörg Widmer, and Martin Mauve. Hierarchical Location Service for Mobile Ad Hoc Networks. *ACM SIGMOBILE Mobile Computing and Communications Review*, 8(4):47–58, 2004.

[84] Richard Knoblauch, Martin Pietrucha, and Marsha Nitzburg. Field Studies of Pedestrian Walking Speed and Start-Up Time. *Transportation Research Record*, 1538:27–38, 1996.

[85] Young-Bae Ko and Nitin H. Vaidya. Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms. In *Second IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[86] Young-Bae Ko and Nitin H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. *Wireless Networks*, 6(4):307–321, 2000.

[87] Young-Bae Ko and Nitin H. Vaidya. GeoTORA: a Protocol for Geocasting in Mobile Ad Hoc Networks. In *International Conference on Network Protocols*, 2000.

[88] André König. *Security in Infrastrure-less and Decentralized Communication Networks*. PhD thesis, Technische Universität Darmstadt, 2011.

[89] Dionysios Kostoulas, Dimitrios Psaltoulis, Indranil Gupta, Kenneth P. Birman, and Alan Demers. Active and Passive Techniques for Group Size Estimation in Large-Scale and Dynamic Distributed Systems. *Journal of Systems and Software*, 80(10):1639–1658, 2007.

[90] Aleksandra Kovacevic. *Peer-to-Peer Location-Based Search: Engineering a Novel Peer-to-Peer Overlay Network*. PhD thesis, Technische Universität Darmstadt, 2009.

[91] Aleksandra Kovacevic, Sebastian Kaune, Patrick Mukherjee, Nicolas Liebau, and Ralf Steinmetz. Benchmarking Platform for Peer-to-Peer Systems. *it - Information Technology (Methods and Applications of Informatics and Information Technology)*, 49(5):312–319, 2007.

[92] Evangelos Kranakis, Harvinder Singh, and Jorge Urrutia. Compass Routing on Geometric Networks. In *Proceedings of the 11th Canadian Conference on Computational Geometry*, 1999.

[93] Petteri Kuosmanen. Classification of Ad Hoc Routing Protocols. Technical report, Finnish Defence Forces, Naval Academy, Finland, 2002.

[94] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET Simulation Studies: the Incredibles. *ACM SIGMOBILE Mobile Computer Communication Review*, 9(4):50–61, 2005.

[95] Stuart Kurkowski, William C. Navidi, and Tracy Camp. Constructing MANET Simulation Scenarios That Meet Standards. In *IEEE Internatonal Conference on Mobile Adhoc and Sensor Systems*, 2007.

[96] James F. Kurose and Keith W. Ross. *Computer Networking: a Top-Down Approach*. Addison-Wesley Publishing Company, 6th edition, 2012.

[97] Kyungman Kwak, Gonzalo Huerta-Canepa, Yangwoo Ko, Dongman Lee, and Soon J. Hyun. An Overlay-Based Resource Monitoring Scheme for Social Applications in MANET. In *Proceedings of the 33rd Annual IEEE International Computer Software and Applications Conference*, 2009.

[98] Stephen Lawson. Survey: Wireless Networks Are Near Capacity, 2011. URL `http://www.pcworld.com/article/235964/wireless_networks_near_capacity.html`. Last accessed on July 18, 2014.

[99] Suk-Bok Lee, Starsky H. Y. Wong, Kang-Won Lee, and Songwu Lu. Content Management in a Mobile Ad Hoc Network: Beyond Opportunistic Strategy. In *Proceedings of the 30th IEEE International Conference on Computer Communications*, 2011.

[100] Ji Li, Karen Sollins, and Dah-Yoh Lim. Implementing Aggregation and Broadcast over Distributed Hash Tables. *Computer Communications*, 35(1):81–92, 2005.

[101] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A Scalable Location Service for Geographic Ad Hoc Routing. In *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, 2000.

[102] Jin Liang, Xiaohui Gu, and Klara Nahrstedt. Self-Configuring Information Management for Large-Scale Service Overlays. In *Proceedings of the 26th IEEE International Conference on Computer Communications*, 2007.

[103] Wen-Hwa Liao, Yu-Chee Tseng, Kuo-Lun Lo, and Jang-Ping Sheu. GeoGRID: a Geocasting Protocol for Mobile Ad Hoc Networks Based on GRID. *Journal of Internet Technology*, 1(2):23–32, 2000.

[104] Nicolas Liebau. *Trusted Accounting in Peer-to-Peer Environments – A Novel Token-Based Accounting Scheme for Autonomous Distributed Systems*. PhD thesis, Technische Universität Darmstadt, 2008.

[105] Justin Lipman, Mehran Abolhasan, Paul Boustead, and Joe Chicharo. An Optimised Resource Aware Approach to Information Collection in Ad Hoc Networks. *Ad Hoc Networks*, 3(5):643–655, 2005.

[106] Dandan Liu, Xiaohua Jia, and Ivan Stojmenović. Quorum and Connected Dominating Sets Based Location Service in Wireless Ad Hoc, Sensor and Actuator Networks. *Computer Communications*, 30(18):3627–3643, 2007.

[107] Donato Macone, Guido Oddi, and Antonio Pietrabissa. MQ-Routing: Mobility-, GPS- and Energy-Aware Routing Protocol in MANETs for Disaster Relief Scenarios. *Ad Hoc Networks*, 11(3):861–878, 2013.

[108] Andrew MacQuire, Andrew Brampton, Idris A. Rai, and Laurent Mathy. Performance Analysis of Stealth DHT with Mobile Nodes. In *4th Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, 2006.

[109] Samuel Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. TAG: a Tiny AGgregation Service for Ad-hoc Sensor Networks. *ACM SIGOPS Operating Systems Review*, 36:131–146, 2002.

[110] Christian Maihofer. A Survey of Geocast Routing Protocols. *IEEE Communications Surveys & Tutorials*, 6(2):32–42, 2004.

[111] Rafik Makhloufi, Grégory Bonnet, Guillaume Doyen, and Dominique Gaïti. Decentralized Aggregation Protocols in Peer-to-Peer Networks : a Survey. In John C. Strassner and Yacine M. Ghamri-Doudane, editors, *Modelling Autonomic Communications Environments*, pages 111–116. Springer, 2009.

[112] Dahlia Malkhi, Moni Naor, and David Ratajczak. Viceroy: a Scalable and Dynamic Emulation of the Butterfly. In *Proceedings of the 21st Annual Symposium on Principles of Distributed Computing*, 2002.

[113] Masoud Mansouri-Samani and Morris Sloman. Monitoring Distributed Systems. *IEEE Network*, 7(6):20–30, 1993.

[114] Matthew L. Massie, Brent N. Chun, and David E. Culler. The Ganglia Distributed Monitoring System: Design, Implementation, and Experience. *Parallel Computing*, 30(7):817–840, 2004.

[115] Martin Mauve, Jörg Widmer, and Hannes Hartenstein. A Survey on Position-Based Routing in Mobile Ad Hoc Networks. *IEEE Network*, 15(6):30–39, 2001.

[116] Petar Maymounkov and David Mazières. Kademlia: a Peer-to-Peer Information System Based on the XOR Metric. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, 2002.

[117] Massimo Mecella, Michele Angelaccio, Alenka Krek, Tiziana Catarci, Berta Buttarazzi, and Schahram Dustdar. WORKPAD: an Adaptive Peer-to-Peer Software Infrastructure for Supporting Collaborative Work of Human Operators in Emergency/Disaster Scenarios. In *International Symposium on Collaborative Technologies and Systems*, 2006.

[118] Shree Murthy and J. J. Garcia-Luna-Aceves. An Efficient Routing Protocol for Wireless Networks. *Mobile Networks and Applications*, 1(2):183–197, 1996.

[119] Soumendra Nanda and David Kotz. Mesh-Mon: a Multi-Radio Mesh Monitoring and Management System. *Special Issue: Modeling, Testbeds, and Applications in Wireless Mesh Networks in Computer Communications*, 31(8):1588–1601, 2008.

[120] William Navidi and Tracy Camp. Stationary Distributions for the Random Waypoint Mobility Model. *IEEE Transactions on Mobile Computing*, 3(1):99–108, 2004.

[121] James Newell and Indranil Gupta. Storia: Time-Indexed Information Monitoring for Large-Scale P2P Networks. Technical report, Departmant of Computer Science, University of Illinois Urbana-Chapaign, 2006.

[122] Anastasios Noulas, Salvatore Scellato, Cecilia Mascolo, and Massimiliano Pontil. An Empirical Study of Geographic User Activity Patterns in Foursquare. In *International AAAI Conference on Weblogs and Social Media*, 2011.

[123] Erik Nygren, Ramesh K. Sitaraman, and Jennifer Sun. The Akamai Network: a Platform for High-Performance Internet Applications. *ACM SIGOPS Operating Systems Review*, 44(3):2–19, 2010.

[124] Alex Olshevsky and John N. Tsitsiklis. Convergence Rates in Distributed Consensus and Averaging. In *45th IEEE Conference on Decision and Control*, 2006.

[125] Andy Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly & Associates, 2001.

[126] Jörg Ott, Esa Hyytiä, Pasi Lassila, Tobias Vaegs, and Jussi Kangasharju. Floating Content: Information Sharing in Urban Areas. In *IEEE International Conference on Pervasive Computing and Communications*, 2011.

[127] Panagiotis Pantazopoulos, Ioannis Stavrakakis, Andrea Passarella, and Marco Conti. Efficient Social-Aware Content Placement in Opportunistic Networks. In *7th International Conference on Wireless On-Demand Network Systems and Services*, 2010.

[128] Kyoung Soo Park and Vivek S. Pai. CoMon: a Mostly-Scalable Monitoring System for PlanetLab. *ACM SIGOPS Operating Systems Review*, 40(1):65–74, 2006.

[129] Vincent D. Park and M. Scott Corson. A Highly Adaptive Distributed Routing Algorithm for Mobile Wireless Networks. In *Proceedings of the 16th Annual Joint Conference of the IEEE Computer and Communications Societies*, 1997.

[130] Diego Passos, Douglas V. Teixeira, Débora C. Muchaluat-Saade, Luiz C. S. Magalhães, and Célio V. N. Albuquerque. Mesh Network Performance Measurements. In *5th International Information and Telecommunication Technologies Symposium*, 2006.

[131] Guangyu Pei, Mario Gerla, and Tsu-Wei Chen. Fisheye State Routing: a Routing Scheme for Ad Hoc Wireless Networks. In *IEEE International Conference on Communications*, 2000.

[132] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the Conference on Communications Architectures, Protocols and Applications*, 1994.

[133] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc On-Demand Distance Vector Routing. In *2nd IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[134] Anna-Kaisa Pietiläinen, Earl Oliver, Jason LeBrun, George Varghese, and Christophe Diot. MobiClique: Middleware for Mobile Social Networking. In *Proceedings of the 2nd ACM Workshop on Online Social Networks*, 2009.

[135] C. Greg Plaxton, Mitul Tiwari, and Praveen Yalagandula. Online Aggregation over Trees. In *IEEE International Parallel and Distributed Processing Symposium*, 2007.

[136] Konstantin Pussep. *Peer-Assisted Video-on-Demand: Cost Reduction and Performance Enhancement for Users, Overlay Providers, and Network Operators*. PhD thesis, Technische Universität Darmstadt, 2011.

[137] Krishna N. Ramachandran, Elizabeth M. Belding-Royer, and Kevin C. Almeroth. DAMON: a Distributed Architecture for Monitoring Multi-Hop Mobile Networks. In *Proceedings of the IEEE Communications Conference on Sensor and Ad Hoc Communications and Networks*, 2004.

[138] Ram Ramanathan and Jason Redi. A Brief Overview of Ad Hoc Networks: Challenges and Directions. *IEEE Communications Magazine*, 40(5):20–22, 2002.

[139] Imran Rao, Aaron Harwood, and Shanika Karunasekera. A Gossip-Based Asynchronous Aggregation Protocol for P2P Systems. In *Proceedings of the 35th IEEE Conference on Local Computer Networks*, 2010.

[140] Thomas Repantis, Jeff Cohen, Scott Smith, and Joel Wein. Scaling a Monitoring Infrastructure for the Akamai Network. *ACM SIGOPS Operating Systems Review*, 44(3):20–26, 2010.

[141] Björn Richerzhagen and Ralf Steinmetz. Towards an Adaptive Publish/Subscribe Approach Supporting Transitions. In Guillaume Doyen, Martin Waldburger, Pavel Čeleda, Anna Sperotto, and Burkhard Stiller, editors, *Emerging Management Mechanisms for the Future Internet*, pages 84–87. Springer, 2013.

[142] Roberto Riggio, Matteo Gerola, Daniele Miorandi, Andrea Zanardi, and Francois Jan. A Distributed Network Monitoring Framework for Wireless Networks. In *IFIP/IEEE International Symposium on Integrated Network Management*, 2011.

[143] Antony I. T. Rowstron and Peter Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms*, 2001.

[144] Elizabeth M. Royer and Chai-Keong Toh. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. *IEEE Personal Communications*, 6 (2):46–55, 1999.

[145] Stephan Rührup. Theory and Practice of Geographic Routing. In Hai Liu, Xiaowen Chu, and Yiu-Wing Leung, editors, *Ad Hoc and Sensor Wireless Networks: Architectures, Algorithms and Protocols*, pages 69–88. Bentham Science, 2009.

[146] Karsten Saller, Kamill Panitzek, and Max Lehn. Benchmarking Methodology. In Wolfgang Effelsberg, Ralf Steinmetz, and Thorsten Strufe, editors, *Benchmarking Peer-to-Peer Systems*, pages 19–45. Springer, 2013.

[147] Sandvine. Global Internet Phenomena Report 1H 2014, 2014. URL `https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf`. Last accessed on July 18, 2014.

[148] Rüdiger Schollmeier, Ingo Gruber, and Florian Niethammer. Protocol for Peer-to-Peer Networking in Mobile Environments. In *Proceedings of the 12th International Conference on Computer Communications and Networks*, 2003.

[149] Pablo Serrano, Michael Zink, and James F. Kurose. Assessing the Fidelity of COTS 802.11 Sniffers. In *Proceedings of the 28th IEEE International Conference on Computer Communications*, 2009.

[150] Muhammad Z. Shafiq, Lusheng Ji, Alex X. Liu, Jeffrey Pang, Shobha Venkataraman, and Jia Wang. A First Look at Cellular Network Performance During Crowded Events. In *Proceedings of the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2013.

[151] Chien-Chung Shen, Chavalit Srisathapornphat, and Chaiporn Jaikaeo. An Adaptive Management Architecture for Ad Hoc Networks. *IEEE Communications Magazine*, 41(2):108–115, 2003.

[152] John P. Snyder. Map Projections: a Working Manual. Technical report, U.S. Geological Survey, 1987.

[153] Sarah Spiekermann. General Aspects of Location-Based Services. In Jochen Schiller and Agnès Voisard, editors, *Location-Based Services*, pages 9–26. Morgan Kaufmann, 2004.

[154] Ralf Steinmetz and Klaus Wehrle. What is This "Peer-to-Peer" About? In Ralf Steinmetz and Klaus Wehrle, editors, *Peer-to-Peer Systems and Applications*, pages 9–16. Springer, 2005.

[155] Dominik Stingl, Christian Gross, Julius Rückert, Leonhard Nobach, Aleksandra Kovacevic, and Ralf Steinmetz. PeerfactSim.KOM: a Simulation Framework for Peer-to-Peer Systems. In *Proceedings of the International Conference on High Performance Computing and Simulation*, 2011.

[156] Dominik Stingl, Christian Gross, Karsten Saller, Sebastian Kaune, and Ralf Steinmetz. Benchmarking Decentralized Monitoring Mechanisms in Peer-to-Peer Systems. In *Proceedings of the Joint WOSP/SIPEW International Conference on Performance Engineering*, 2012.

[157] Dominik Stingl, Christian Gross, Leonhard Nobach, Ralf Steinmetz, and David Hausheer. BlockTree: Location-Aware Decentralized Monitoring in Mobile Ad Hoc Networks. In *Proceedings of the 38th IEEE Conference on Local Computer Networks*, 2013.

[158] Dominik Stingl, Christian Gross, and Karsten Saller. Decentralized Monitoring in Peer-to-Peer Systems. In Wolfgang Effelsberg, Ralf Steinmetz, and Thorsten Strufe, editors, *Benchmarking Peer-to-Peer Systems*, pages 81–113. Springer, 2013.

[159] Dominik Stingl, Björn Richerzhagen, Fabio Zöllner, Christian Gross, and Ralf Steinmetz. PeerfactSim.KOM: Take it Back to the Streets. In *Proceedings of the International Conference on High Performance Computing and Simulation*, 2013.

[160] Dominik Stingl, Reimond Retz, Björn Richerzhagen, Christian Gross, and Ralf Steinmetz. Mobi-G: Gossip-Based Monitoring in MANETs. In *IEEE/IFIP Network Operations and Management Symposium*, 2014.

[161] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: a Scalable Peer-To-Peer Lookup Service for Internet Applications. In *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, 2001.

[162] Ivan Stojmenović. Position-Based Routing in Ad Hoc Networks. *IEEE Communications Magazine*, 40(7):128–134, 2002.

[163] Ivan Stojmenović and Xu Lin. Loop-Free Hybrid Single-Path/Flooding Routing Algorithms with Guaranteed Delivery for Wireless Networks. *IEEE Transactions on Parallel and Distributed Systems*, 12(10):1023–1032, 2001.

[164] Ivan Stojmenović, Anand P. Ruhil, and D. K. Lobiyal. Voronoi Diagram and Convex Hull Based Geocasting and Routing in Wireless Networks. In *8th IEEE International Symposium on Computers and Communication*, 2003.

[165] Daniel Stutzbach and Reza Rejaie. Understanding Churn in Peer-to-Peer Networks. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, 2006.

[166] Hideaki Takagi and Leonard Kleinrock. Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals. *IEEE Transactions on Communications*, 32(3):246–257, 1984.

[167] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks*. Pearson Prentice Hall, 2010.

[168] Paul Taylor. Data Overload Threatens Mobile Networks, 2012. URL `http://www.ft.com/intl/cms/s/0/caeb0766-9635-11e1-a6a0-00144feab49a.html#axzz35vYGZtDY`. Last accessed on July 18, 2014.

[169] Wesley W. Terpstra, Jussi Kangasharju, Christof Leng, and Alejandro P. Buchmann. Bubblestorm: Resilient, Probabilistic, and Exhaustive Peer-to-Peer Search. *ACM SIGCOMM Computer Communication Review*, 37(4):49–60, 2007.

[170] Wesley W. Terpstra, Christof Leng, and Alejandro P. Buchmann. Brief Announcement: Practical Summation via Gossip. In *Proceedings of the 26th Annual Symposium on Principles of Distributed Computing*, 2007.

[171] Chai-Keong Toh. A Novel Distributed Routing Protocol To Support Ad-Hoc Mobile Computing. In *Conference Proceedings of the 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications*, 1996.

[172] Yu-Chee Tseng, Sze-Yao Ni, Yuh-Shyan Chen, and Jang-Ping Sheu. The Broadcast Storm Problem in a Mobile Ad Hoc Network. *Wireless Networks*, 8(2): 153–167, 2002.

[173] Cristian Tuduce and Thomas Gross. Resource Monitoring Issues in Ad Hoc Networks. In *International Workshop on Wireless Ad-Hoc Networks*, 2004.

[174] Ruud van de Bovenkamp, Fernando Kuipers, and Piet Van Mieghem. Gossip-Based Counting in Dynamic Networks. In *Proceedings of the 11th International IFIP TC 6 Conference on Networking*, 2012.

[175] Robbert van Renesse and Adrian Bozdog. Willow: DHT, Aggregation, and Publish/Subscribe in one Protocol. In *Proceedings of the 3rd International Conference on Peer-to-Peer Systems*, 2004.

[176] Robbert van Renesse, Kenneth P. Birman, and Werner Vogels. Astrolabe: a Robust and Scalable Technology for Distributed System Monitoring, Management, and Data Mining. *ACM Transactions on Computer Systems*, 21(2):164–206, 2003.

[177] Tarmo Virki and Nicola Leske. Networks Desperately Seek Data Capacity, 2010. URL `http://www.reuters.com/article/2010/02/17/us-mobile-fair-networks-idUSTRE61G3EY20100217`. Last accessed on July 18, 2014.

[178] Kirsi Virrantaus, Jouni Markkula, Artem Garmash, Vagan Terziyan, Jari Veijalainen, Artem Katanosov, and Henry Tirri. Developing GIS-Supported Location-Based Services. In *Proceedings of the 2nd International Conference on Web Information Systems Engineering*, 2001.

[179] Feng Wang, Yongqiang Xiong, and Jiangchuan Liu. mTreebone: a Collaborative Tree-Mesh Overlay Network for Multicast Video Streaming. *IEEE Transactions on Parallel and Distributed Systems*, 21(3):379–392, 2010.

[180] Fetahi Wuhib and Rolf Stadler. Adaptive Real-time Monitoring in Mobile Wireless Networks. Technical report, Royal Institute of Technology, January 2008.

[181] Fetahi Wuhib, Mads Dam, Rolf Stadler, and Alexander Clem. Robust Monitoring of Network-Wide Aggregates Through Gossiping. *IEEE Transactions on Network and Service Management*, 6(2):95–109, 2009.

[182] Praveen Yalagandula and Mike Dahlin. A Scalable Distributed Information Management System. *ACM SIGCOMM Computer Communication Review*, 34(4): 379–390, 2004.

[183] Praveen Yalagandula and Mike Dahlin. Shruti: a Self-Tuning Hierarchical Aggregation System. In *International Conference on Self-Adaptive and Self-Organizing Systems*, 2007.

[184] Thomas Zahn and Jochen Schiller. Designing Structured Peer-to-Peer Overlays as a Platform for Distributed Network Applications in Mobile Ad Hoc Networks. *Computer Communications*, 31(3):643–654, 2008.

[185] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, Zhaoguang Wang, Robert P. Dick, Z. Morley Mao, and Lei Yang. Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones. In *Proceedings of the IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, 2010.

[186] Kathryn Zickuhr. Location-Based Services, 2013. URL `http://www.pewinternet.org/~/media//Files/Reports/2013/PIP_Location-based%20services%202013.pdf`. Last accessed on July 18, 2014.

[187] Michael Zink and Andreas Mauthe. P2P Streaming using Multiple Description Coded Video. In *Proceedings of the 30th Euromicro Conference*, 2004.

[188] Michael Zink, Kyoungwon Suh, Yu Gu, and James F. Kurose. Characteristics of YouTube Network Traffic at a Campus Network – Measurements, Models, and Implications. *Computer Networks*, 53(4):501–514, 2009.

[189] Michele Zorzi and Ramesh R. Rao. Geographic Random Forwarding (GeRaF) for Ad Hoc and Sensor Networks: Energy and Latency Performance. *IEEE Transactions on Mobile Computing*, 2(4):349–365, 2003.

A

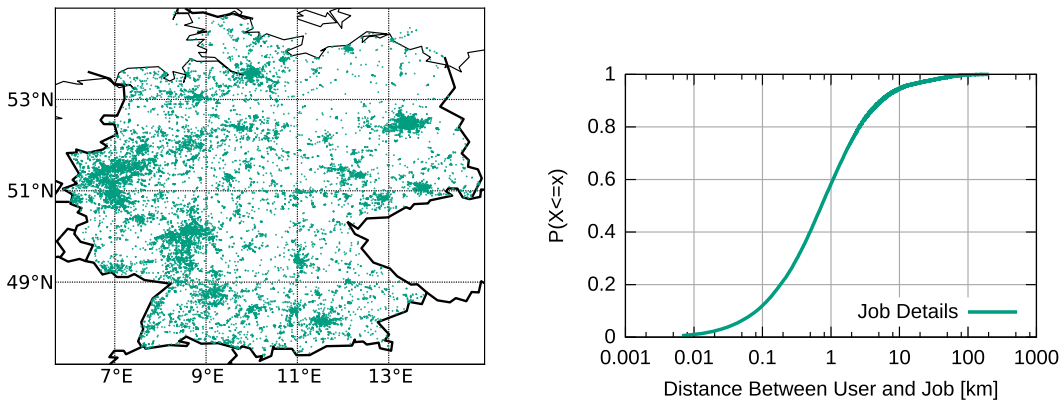
APPENDIX

A.1 LOCATION-BASED SERVICES: A CASE STUDY

For a better understanding of the characteristics of location-based services, specifically, regarding the relation between a user and its content of interest, we analyzed an existing location-based service. The analysis is based on data that we obtained from the crowd sourcing application *AppJobber*, which was provided by the "wer denkt was GmbH". In AppJobber users are able to query their environment for so-called micro jobs, which represent small tasks that are rewarded on completion. Possible tasks range from validating specific geo-information by taking pictures with a smartphone to answering questionnaires. If a user is interested in an overview about available jobs in a certain location, it queries this location and obtains a set of so-called *Job Requests*, which briefly describe available jobs at their location. If a user wants to know more about a certain job, it retrieves the details of this job, which are denoted as *Job Details*.

The underlying data set for the analysis of AppJobber was collected for Germany over one year from December 2011 until December 2012. During this time more than 13 million requests have been processed, comprising 12,993,444 Job Requests and 33,204 Job Details, while 30,694 jobs have been completed. Figure 81a depicts the geographical distribution of the requested Job Details over Germany to provide an overview where AppJobber has been used in Germany during that time.

To analyze the relation between a user and the relevant content we evaluated the distance between a user's current position and the position of the queried Job Details. Figure 81b depicts the distribution of the distance for the queried Job Details. The results indicate that approximately 12 % of all Job Details exhibit a distance below 100 m and 58.3 % of the retrieved details are less than one kilometer away. Only



(a) Geographical distribution of the requested Job Details over Germany

(b) Distribution of the distance between a user and the requested Job Details

Figure 81: Characteristics of requested Job Details from the crowd sourcing application AppJobber

6.4 % of the users retrieved Job Details that are more than 10 km away, coming to a maximum distance of 201.4 km between a user and a Job Detail. Based on the example of AppJobber, the results indicate that a user is highly interested in its direct vicinity and primarily retrieves content and information about nearby locations.

A.2 BLOCKTREE.KOM: EVALUATION OF ADDITIONAL SYSTEM PARAMETERS

In this section we evaluate the remaining system parameters of BlockTree.KOM, which have not been examined in the evaluation chapter. Table 44 lists the corresponding system parameters that will be investigated in this section. The configuration of the fixed system parameters corresponds to the default configuration of P- and C-BlockTree.KOM, as introduced in Section 6.3.2.1. Before the examination of the system parameters, we take a closer look on the different traffic types and quantify the corresponding amount.

Fixed system parameter	**Configuration**
Estimated maximum communication range	205 m
Maximum forwarding time	400 ms
Position refresh interval	10 s
Block length	145 m
Sector edge length of C-BlockTree.KOM	3
Sector edge length of P-BlockTree.KOM	2
Maximum hierarchy level of C-BlockTree.KOM	2
Maximum hierarchy level of P-BlockTree.KOM	4
Aggregation interval	15 s
Varied system parameter	**Configuration**
Leaf information timeout factor	1.2
Timeout factor	1.6
Blocking interval factor	0.9
Operation blocking interval factor	1

Table 44: Overview on the default configuration of the fixed and varied system parameters for the evaluation of the remaining system parameters of C- and P-BlockTree.KOM

A.2.1 *Evaluation of the Different Traffic Types of P- and C-BlockTree.KOM*

For the evaluation and a ranking of the different traffic types we rely on the obtained results from the scenarios with and without churn, which are presented in Section 6.3.2.2. Table 45 and 46 show the overall traffic and outline its partitioning into the different traffic types per approach. The corresponding figures represent the mean upload and download traffic per second per node for the respective traffic types. The arising traffic of P-BlockTree.KOM just consists of Leaf-Broadcasting and Aggregating-Up traffic, because the approach neither relies on a separate result dissemination nor on a separate mechanism for the replication of data.

Once again, the listed overall mean upload and download traffic outline that P-BlockTree.KOM generates a higher amount of traffic than C-BlockTree.KOM. The results of P-BlockTree.KOM indicate that a large fraction arises from the transmission and reception of Aggregating-Up messages due to the staggered execution

	P-BlockTree.KOM		C-BlockTree.KOM	
Traffic type [Bps]	**No churn**	**Churn**	**No churn**	**Churn**
Overall traffic	71.07	69.44	45.56	46.29
LEAF-BROADCASTING traffic	8.05	8.06	7.21	7.19
AGGREGATING-UP traffic	63.03	61.39	22.47	23.26
DISSEMINATING-DOWN traffic	–	–	8.95	8.72
Replication traffic	–	–	6.94	7.11

Table 45: Classification of the overall arising upload traffic into LEAF-BROADCASTING, AGGREGATING-UP, DISSEMINATING-DOWN, and replication traffic

	P-BlockTree.KOM		C-BlockTree.KOM	
Traffic type [Bps]	**No churn**	**Churn**	**No churn**	**Churn**
Overall traffic	1852.95	1971.93	1322.53	1460.80
LEAF-BROADCASTING traffic	229.13	252.01	206.91	226.04
AGGREGATING-UP traffic	1623.82	1719.92	690.12	767.58
DISSEMINATING-DOWN traffic	–	–	212.08	222.47
Replication traffic	–	–	213.43	244.72

Table 46: Classification of the overall arising download traffic into LEAF-BROADCASTING, AGGREGATING-UP, DISSEMINATING-DOWN, and replication traffic

of the corresponding operations, which are triggered for every level during an aggregation interval. Taking the download traffic as an example, the resulting mean AGGREGATING-UP traffic represents 87.63 % and 87.22 % of the overall download traffic without and with churn. In terms of C-BLOCKTREE.KOM, the AGGREGATING-UP traffic only accounts for 52.18 % and 52.55 % without and with churn, because the corresponding AGGREGATING-UP operation is executed only once per aggregation interval. For P-BLOCKTREE.KOM, the remaining traffic arises due the transmission of LEAF-BROADCASTING messages. In this context, the results reveal that C-BLOCKTREE.KOM generates less traffic for the dissemination of LEAF-BROADCASTING information, though, both approaches use the same strategy and interval to execute the operation. The reason for the decreased LEAF-BROADCASTING traffic of C-BLOCKTREE.KOM results from the fact that a node at l_x with $x > 0$ uses a replication message (i) to replicate its hierarchy table as well as (ii) to disseminate its locally measured attributes in the block. Consequently, only nodes at l_0 exclusively use LEAF-BROADCASTING messages to disseminate their local information, which leads to the illustrated difference between both approaches for the LEAF-BROADCASTING traffic. The remaining traffic of C-BLOCKTREE.KOM is nearly evenly divided into DISSEMINATING-DOWN and replication traffic. Despite the fact that only a fraction of nodes transmits and receives replication messages, the amount of the replication traffic is comparable to the DISSEMINATING-DOWN traffic. This outcome indicates that the replication

of the hierarchy table constitutes a costly operation, which is reflected by the transmission of large messages but which are indispensable for the proper operation of the hierarchical approach.

A.2.2 *Leaf Information Timeout Factor*

The leaf information timeout factor represents a factor, which is multiplied with the aggregation interval to determine the effective duration of the leaf information timeout $t_{leafInfoTimeout}$. After the specified amount of time, cached data of neighboring nodes from the same block are removed from the leaf information table. Thus, the parameter just influences a node's current view of the state in its block but does not affect the communication between the nodes, which will become apparent by the resulting cost in terms of traffic.

Table 47 provides an overview about the different configurations of the system parameter. A factor of 1.2 represents the default configuration, which yields to a longer leaf information timeout than the aggregation interval. Consequently, the leaf information table is only purged from old data, if a neighboring node could not update the data during its previous aggregation interval. With a reduced leaf information timeout factor we evaluate how both approaches perform if the table is cleaned before neighboring nodes are able to send their periodical updates. Vice versa, we assess the performance if the leaf information timeout is extended and more than twice as long as the aggregation interval.

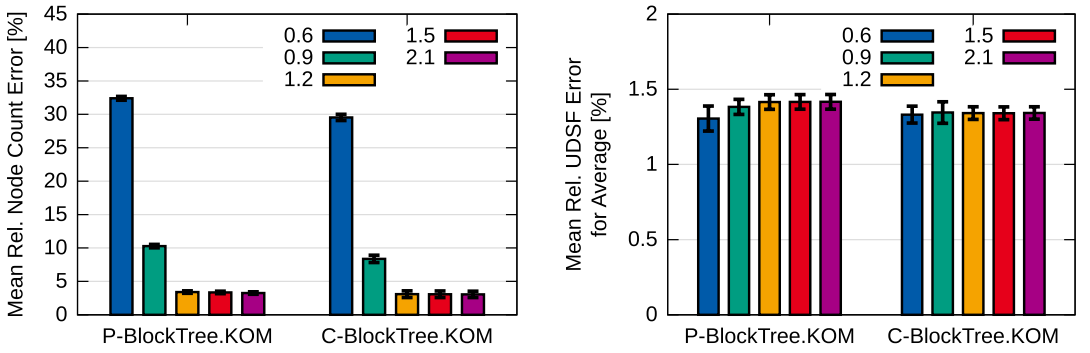
System parameter	**Configuration**
Leaf information timeout factor	0.6, 0.9, <u>1.2</u>, 1.5, 2.1

Table 47: System parameter variation of the leaf information timeout factor. The underlined value represents the default configuration of the system parameter.

ACCURACY AND STALENESS

Figure 82a depicts the mean relative node count error as a function of the varied leaf information timeout factor and reveals the negative impact of short factors. Configurations of the parameter, which lead to a shorter leaf information timeout than the aggregation interval, considerably increase the mean relative node count error, because a shorter timeout often purges the leaf information table before an AGGREGATING-UP operation is triggered by the aggregation interval. As a result the AGGREGATING-UP operation uses data that do not correctly reflect the current state in the corresponding block, leading to the observed mean relative error. A longer leaf information timeout than the aggregation interval has no impact on accuracy, because the data are either timely updated before the next AGGREGATING-UP operation or the corresponding data of a node, which left the current block, are actively purged, as described in Section 4.4.3 and 4.5.3 for C- and P-BLOCKTREE.KOM.

Figure 82b shows the mean relative UDSF error and outlines the small impact of the factor on the mean relative monitoring error of this attribute. The reason for the small effect on the relative error mainly arises from the application of the average as aggregation function. Based on this function, the results reveal that both approaches provide accurate results, because the aggregation function mitigates the impact of



(a) Mean relative node count error

(b) Mean relative UDSF error using the average as aggregation function

Figure 82: Overview on accuracy of the provided monitoring results for a varying leaf information timeout factor

missing data. In contrast, the monitoring accuracy of this attribute mainly depends on the time to collect and disseminate the data, which is represented by staleness and depicted in Figure 83. In fact, the presented results outline that a faster collection and dissemination as well as the integration of recent data lead to a reduced staleness and a reduced mean relative UDSF error for a shorter interval. The reduced staleness for shorter leaf information timeouts originates from the exclusive utilization of recent data for the AGGREGATING-UP operation. Older data are purged from the leaf information table during every aggregation interval so that data are collected, which have been stored in the table just before executing the AGGREGATING-UP operation. If the leaf information timeout exceeds the aggregation interval, the staleness of the results is not influenced anymore. This originates from the fact that the leaf information table is periodically updated based on the aggregation interval or that stale entries are deleted if neighboring nodes leave the current block.

TRAFFIC

As already indicated in the introduction and confirmed by the depicted results (cf. Figure 84a and 84b) the leaf information timeout factor does not influence the arising upload and download traffic. Due to the fact that every node operates at least on its locally measured data and initiates an AGGREGATING-UP operation, the periodic

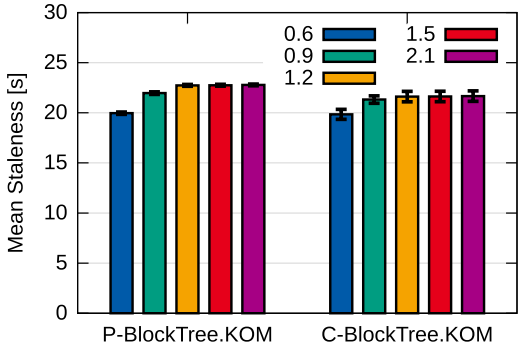
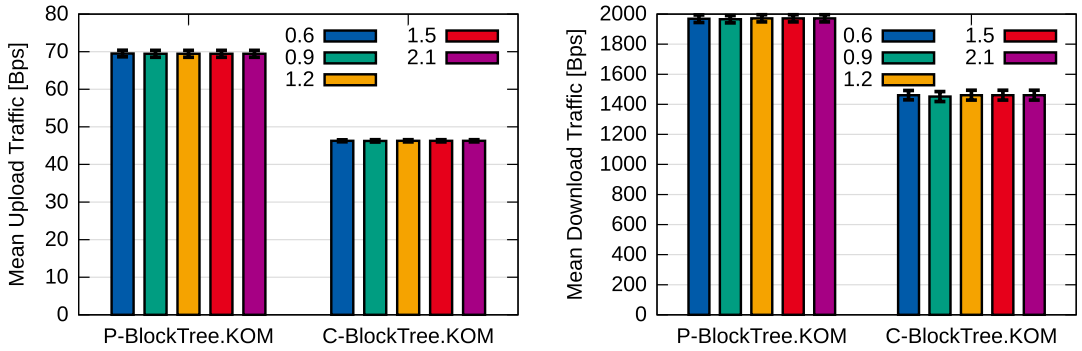


Figure 83: Mean staleness of the provided monitoring results for a varying leaf information timeout factor



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 84: Overview on the overall mean traffic per node for a varying leaf information timeout factor

operations will always trigger an information exchange, independent of the current configuration of the leaf information timeout factor. Consequently and as outlined by the results, the traffic is not influenced.

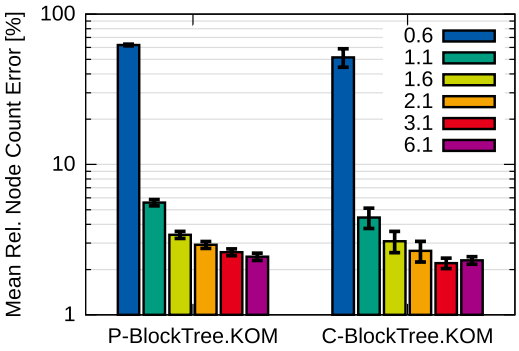
A.2.3 *Timeout Factor*

Similar to the previous factor timeout factor represents a parameter, which does not influence the communication between nodes. Instead, it specifies when to purge data from the tables, which hold the current knowledge of a node. In this context, the timeout factor specifies when to clean the hierarchy or result table. Consequently, we mainly focus on performance during the following evaluation and just examine the replication traffic of C-BlockTree.KOM. Apart from that, the parameter has no influence on the arising cost.

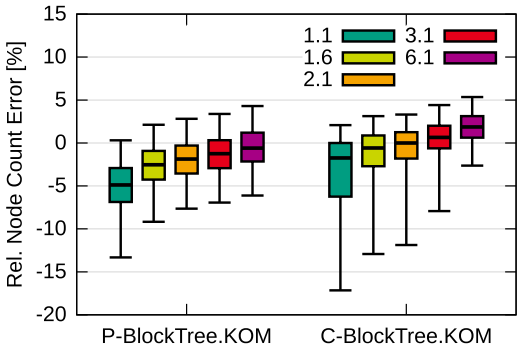
The timeout factor just represents a factor, which is multiplied with the aggregation interval. The computed interval determines the duration of the Aggregating-Up and the Disseminating-Down timeout $t_{aggUpTimeout}$ and $t_{dissDownTimeout}$ in terms of C-BlockTree.KOM as well as the Aggregating-Up and the result caching timeout $t_{aggUpTimeout}$ and $t_{resCacheTimeout}$ for P-BlockTree.KOM. The different values for the configuration of the timeout factor are listed in Table 48. The parameter is varied in both directions to examine the impact of smaller and larger intervals on performance and cost. In contrast to the experiments of the $2^k * r$ factorial design we extend the set of configurations with 0.6 and 6.1 to stress the impact of very small and large factors. With 1.1, 2.1, 3.1, and 6.1 the corresponding timeouts are longer than the n-fold of the aggregation interval to evaluate the impact of purging the tables after an aggregation interval has been completed. A factor of 0.6 and 1.6 is chosen to observe the impact if the corresponding tables are purged during an aggregation interval.

System parameter	Configuration
Timeout factor	0.6, 1.1, <u>1.6</u>, 2.1, 3.1, 6.1

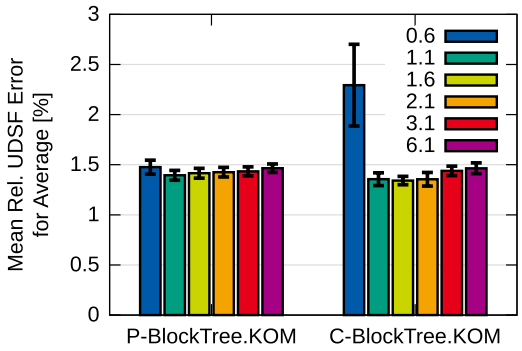
Table 48: System parameter variation of the timeout factor. The underlined value represents the default configuration of the system parameter.



(a) Mean relative node count error



(b) Distribution of the relative node count error of all collected measurements. The plot does not comprise the results for a timeout factor of 0.6 to highlight the differences between the remaining results.

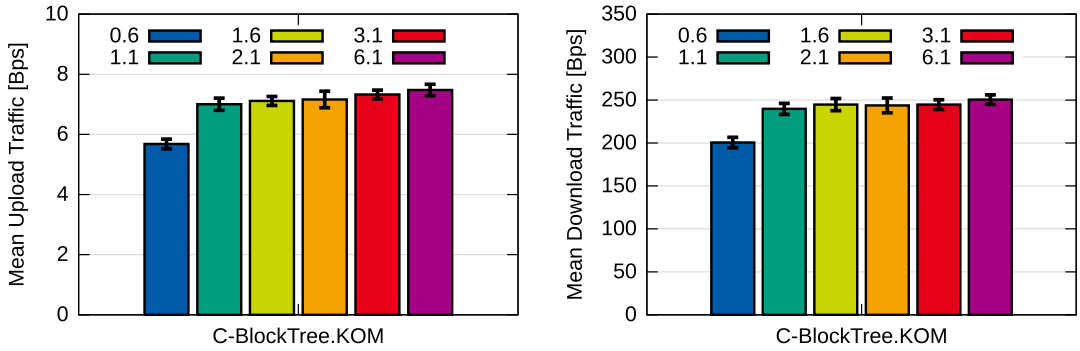


(c) Mean relative UDSF error using the average as aggregation function

Figure 85: Overview on accuracy of the provided monitoring results for a varying timeout factor

ACCURACY AND STALENESS

Figure 85 provides an overview about the influence of different configurations of the timeout factor on the accuracy of both approaches. Starting again with the evaluation of the mean relative node count error (cf. Figure 85a), the results reveal the negative impact of a timeout factor below one, which leads to a considerably increased mean relative error. The observed negative outcome for a timeout factor of 0.6 arises from the deletion of data in the hierarchy and result tables before the reception of new information, which is triggered by the next cycle of the aggregation interval and typically updates older results in both tables. After the premature purging of tables before the reception of new information, the tables are nearly empty when starting a new Aggregating-Up or Disseminating-Down operation, which leads to the increased mean relative error. The default configuration of the blocking interval factor, which is set to 0.9 and defines how long messages are blocked, also prohibits a premature transmission of information during the aggregation interval to provide the required information. A reduction of the blocking interval factor counteracts the lack of information, however, the evaluation of this parameter in the following section outlines that the small gain in accuracy and timeliness comes at disproportional cost.



(a) Mean replication upload traffic per node

(b) Mean replication download traffic per node

Figure 86: Overview on the mean replication traffic per node of C-BlockTree.KOM for a varying timeout factor

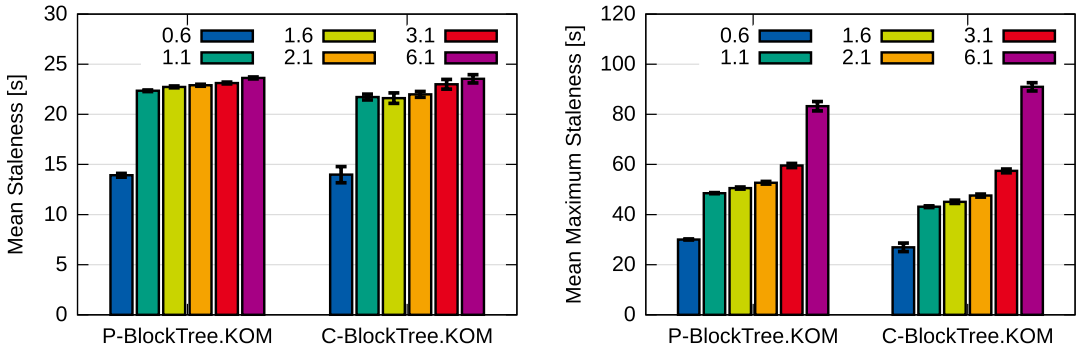
In terms of the node count, the results reveal that a larger timeout factor is beneficial for the accuracy of BlockTree.KOM, because the mean relative error decreases for both approaches. The reason for the improved performance results from the fact that a larger timeout factor avoids an early deletion of data in the hierarchy and result tables and prevents data loss during the collection and dissemination of data over the different levels of the hierarchy.

Figure 85b depicts the distribution of the relative node count error. In this context we omit the results for the first experiment with a timeout factor of 0.6 to focus on the remaining experiments and to stress the differences between them. In fact, the results outline that the accuracy benefits from a growing timeout factor, because it reduces both (i) the deviation of the monitored from the effective state as well as (ii) the fraction of underestimating nodes. Moreover, the results reveal that even large timeout factors, e.g., of 3.1 or 6.1, do not necessarily result in a considerable overestimation, but reduce the deviation from the effective network state and the fraction of underestimating nodes. BlockTree.KOM prevents an overestimation of the results due to the implementation of the following procedures, which are triggered if a node enters a new block (cf. Section 4.4.3 and 4.5.3).

- Nodes purge their leaf information tables of entries from the leaving node.
- To prevent the distribution of incomplete or redundant information a node does neither trigger an Aggregating-Up nor Disseminating-Down operation for a certain amount of time if it detects that it entered a new block.

Based on these procedures, BlockTree.KOM reduces the probability that stale or redundant information is collected and disseminated so that longer timeout intervals are used to reduce the impact of a premature deletion of data in the hierarchy and result tables. However, the results additionally unveil that P-BlockTree.KOM generally suffers from an underestimation even for long timeout intervals, whereas C-BlockTree.KOM begins to overestimate the current state in the network.

Figure 85c depicts the mean relative UDSF error of both approaches of BlockTree.KOM. In terms of P-BlockTree.KOM, the results outline that the applied aggregation function on the attribute mitigates the strong impact of a timeout factor of 0.6 on accuracy. The corresponding mean relative error of this configuration only indicates the tendency that the accuracy decreases. Contrary to this, C-BlockTree.KOM



(a) Mean average staleness of the provided monitoring results

(b) Mean maximum staleness of the provided monitoring results

Figure 87: Overview on different types of staleness of the provided monitoring results for a varying timeout factor

suffers from this configuration of the timeout factor, because the mean relative error significantly increases for a timeout factor of 0.6. To explain this phenomenon we consider the impact on the arising upload and download traffic for the replication of hierarchy tables in the concentrating blocks, as depicted in Figure 86a and 86b. We limit our examination on the replication traffic, because it constitutes the only traffic type, which is influenced by the timeout parameter, while the remaining traffic types are not affected. The results for the replication traffic show that a timeout factor of 0.6 significantly reduces the arising upload and download traffic for the replication, which confirms that the corresponding hierarchy tables in the concentrating blocks do not contain the information from all surrounding blocks or sectors. The missing data in the concentrating blocks cannot be concealed by the applied aggregation function as good as for P-BlockTree.KOM, which explains the increased mean relative UDSF error for a configuration of 0.6.

To complete the evaluation of the timeout factor we examine the impact of the parameter on the mean staleness, as shown in Figure 87a. The results reveal that a larger timeout factor influences the mean staleness and leads to older results. For the special case of a factor 0.6 the results uncover that the mean staleness is considerably reduced, because only recent data is included. However, the observed reduction of staleness comes at the expense of highly incomplete data, which leads to the previously discussed inaccurate results. To stress the impact of a timeout factor on staleness Figure 87b additionally depicts the mean maximum staleness, which is calculated as $t_{stale} = t_{req} - t_{max}$. In this formula t_{max} replaces t_{avg} and constitutes the oldest value, which has been integrated into the global view of the corresponding attribute. Based on this worst case analysis, the results of the mean maximum staleness support the statement for the mean average staleness and confirm that a larger timeout factor generally increases the staleness of the results. However, the results for the mean relative node count and UDSF error indicate that these extreme values have little influence on accuracy, because BlockTree.KOM already deletes the majority of stale and redundant data at the bottom of the hierarchy to prevent their collection and dissemination.

A.2.4 *Blocking Interval Factor*

The blocking interval factor is used to configure both the AGGREGATING-UP and the DISSEMINATING-DOWN blocking interval, which are represented by $t_{aggUpBlocking}$ and $t_{dissDownBlocking}$, respectively. Both intervals specify how long a message ID of an AGGREGATING-UP or DISSEMINATING-DOWN message from a particular block or sector is stored in the message forwarding cache before the ID is removed from the cache. While the cache holds this ID, further messages from the same block or sector are blocked and not forwarded to avoid that the MANET is constantly flooded with messages. Similar to the timeout factor the blocking interval factor just represents a factor, which is multiplied with the aggregation interval to calculate the effective values for $t_{aggUpBlocking}$ and $t_{dissDownBlocking}$. Contrary to the timeout factor the parameter influences the communication between the nodes, because it specifies how long messages are blocked and the corresponding information is ignored.

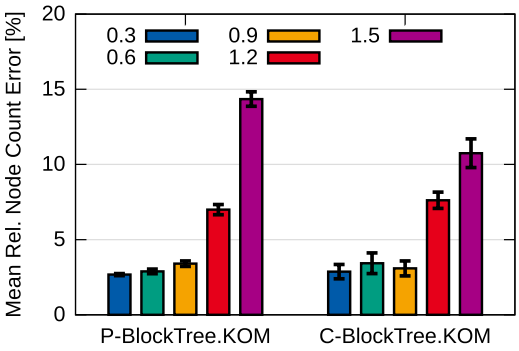
The different configurations of the blocking interval factor are depicted in Table 49. As detailed during the description of the default configuration of BLOCKTREE.KOM a parameter of 0.9 specifies an interval, which blocks most of the messages during an aggregation interval. Before the old aggregation interval has ended, nodes can trigger new AGGREGATING-UP and DISSEMINATING-DOWN operations ahead of schedule. With smaller factors we investigate the influence of nodes, which are not synchronized with the current cycle of the aggregation interval, but trigger their operations in the middle or even at the beginning of the current interval. With larger factors we examine how both approaches perform if the blocking interval of messages exceeds the current aggregation interval. Similar to the variation of the timeout factor we extend the initial configuration range from the experiments of the $2^k * r$ factorial design with 0.3 and 1.5 to stress the impact of very small and large factors.

System parameter	Configuration
Blocking interval factor	0.3, 0.6, <u>0.9</u>, 1.2, 1.5

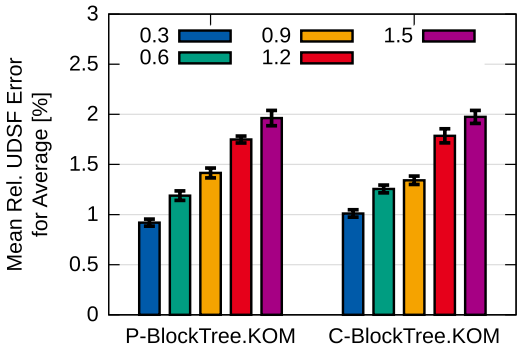
Table 49: System parameter variation of the blocking interval factor. The underlined value represents the default configuration of the system parameter.

ACCURACY AND STALENESS

The impact of different configurations of the blocking interval factor on accuracy of the monitoring results is depicted in Figure 88. In terms of the mean relative node count error (cf. Figure 88a) the results reveal that a configuration with a lower value than the default configuration slightly reduces the mean relative error of P-BLOCKTREE.KOM, whereas no significant tendency is observed for C-BLOCKTREE.KOM. In contrast, a higher factor, which is larger than the factor of the default configuration and defines a longer interval than the aggregation interval, significantly increases the mean relative error for both P- and C-BLOCKTREE.KOM. The increasing error arises from the blocking of AGGREGATING-UP and DISSEMINATING-DOWN messages in a subsequent cycle. If both messages are triggered during the current cycle and the nodes are roughly synchronized, the subsequent transmission of messages is blocked during the next cycle, because the blocking interval is longer than the aggregation interval. Consequently, data collection and result dissemination are delayed,



(a) Mean relative node count error



(b) Mean relative UDSF error using the average as aggregation function

Figure 88: Overview on accuracy of the provided monitoring results for a varying blocking interval factor

as confirmed by the increasing mean staleness in Figure 89. The delayed collection and dissemination lead to the decreasing accuracy in terms of the node count, because the hierarchy and result tables are purged from stale entries, as these entries are not updated in time with recent information. Based on the outcome, it is concluded that the blocking interval factor should be configured so that $t_{aggUpBlocking}$ and $t_{dissDownBlocking}$ are smaller than the aggregation interval. In contrast, a higher timeout factor could be chosen so that the delayed collection and dissemination does not lead to the observed increasing error due to the periodic deletion of stale entries in the hierarchy and result tables. During the following section, we focus on the influence of both parameters in parallel.

Dealing with the impact of the blocking interval factor on the mean relative UDSF error, Figure 88b depicts the corresponding results, which unveil that a smaller factor does not only reduce the mean staleness but the mean relative monitoring error for UDSF as well. A reduction of the factor leads to shorter blocking intervals so that the transmission of messages after the beginning or during a new aggregation interval already improves the accuracy of the results. More precisely, nodes send AGGREGATING-UP and DISSEMINATING-DOWN messages earlier, also leading to a faster data collection and result dissemination. For a longer blocking interval factor the typical and previously discussed characteristics for this attribute are observed.

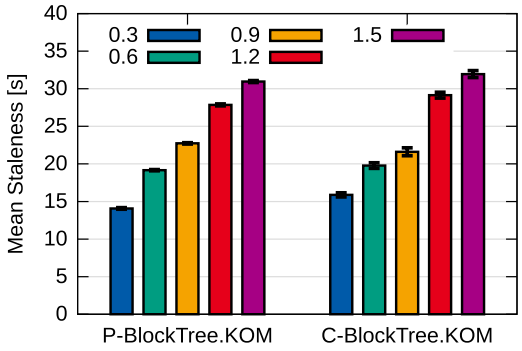
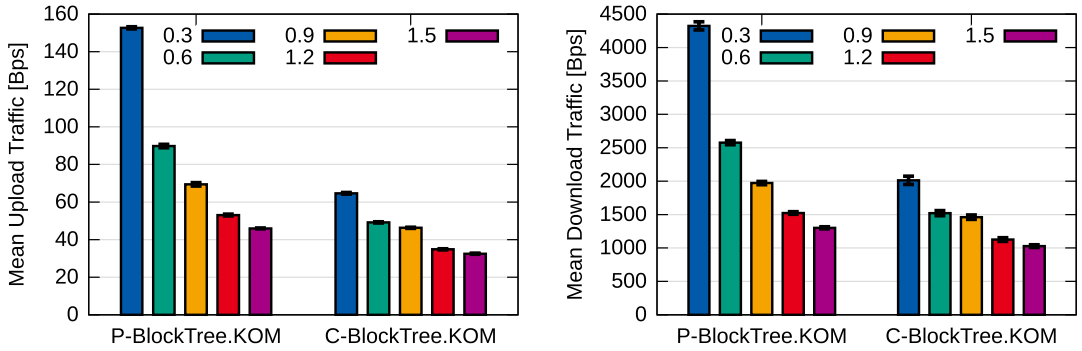


Figure 89: Mean staleness of the provided monitoring results for a varying blocking interval factor



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 90: Overview on the overall traffic per node for a varying blocking interval factor

The mean relative error increases as for the node count, however, the applied aggregation function mitigates the effect of incomplete or lost data on the resulting mean relative error. Consequently, the impact on this attribute is not that strong as for the node count. In fact, the relative UDSF error mainly depends on the duration for the collection of data and the dissemination of results and increases for a decelerated collection and dissemination due to a longer blocking of messages.

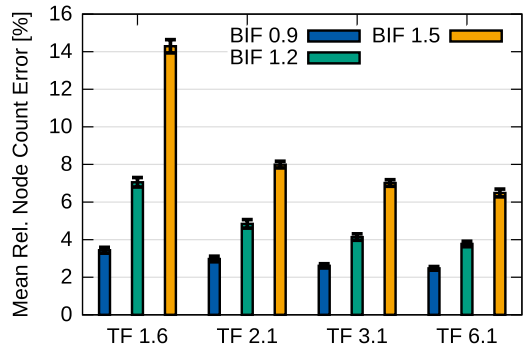
TRAFFIC

Dealing with the resulting upload and download traffic, Figure 90a and 90b uncover that an increasing blocking interval for the transmission of AGGREGATING-UP and DISSEMINATING-DOWN messages leads to a decreasing traffic. Consequently, the higher accuracy in terms of UDSF and the reduced staleness come at increased cost. In terms of the node count a shorter blocking interval factor marginally improves P-BLOCKTREE.KOM's accuracy, whereas no significant improvements are observed for C-BLOCKTREE.KOM. Similar to the observed results for the variation of the aggregation interval (cf. Section 6.3.2.4) the minor gain in performance comes at increasing cost. Since the blocking interval factor only influences the transmission of AGGREGATING-UP and DISSEMINATING-DOWN messages, while the remaining message types remain unaffected, the impact on cost is not that strong as for the aggregation interval. However, the minor performance gain does not justify the observed increase in traffic.

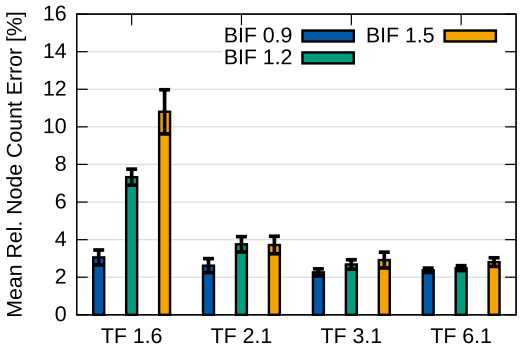
A.2.5 *Evaluating the Combination of Timeout Factor and Blocking Interval Factor*

After the description of the individual parameter variations for the timeout and blocking interval factor, this section contains the combined evaluation of both parameters. The respective configurations for every parameter are depicted in Table 50 and consist of four configurations for the timeout factor and of three configurations for the blocking interval factor, leading to 12 experiments in total.

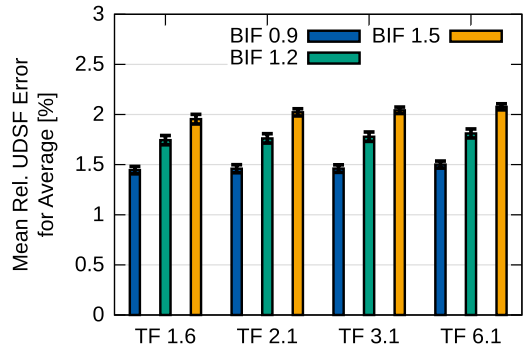
In terms of the timeout factor we restrict the variation to a subset of the initial parameters, because the previously presented results reveal that timeout factors of 0.6 or 1.1 lead to inaccurate and incomplete results (cf. Section A.2.3). Regarding the blocking interval factor, we only rely on values above the default configuration of 0.9, because lower values considerably increase the traffic, while the resulting



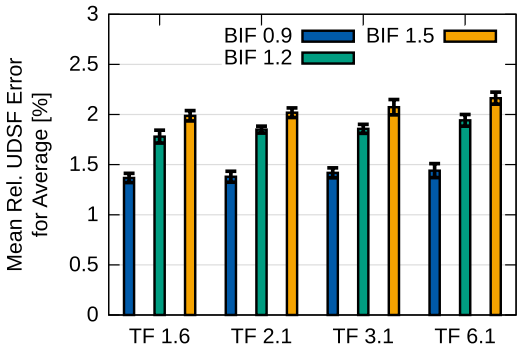
(a) Mean relative node count error of P-BLOCK-TREE.KOM



(b) Mean relative node count error of C-BLOCK-TREE.KOM



(c) Mean relative UDSF error of P-BLOCK-TREE.KOM using the average for aggregation



(d) Mean relative UDSF error of C-BLOCK-TREE.KOM using the average for aggregation

Figure 91: Overview on the mean relative node count and UDSF error for the combined parameter variation

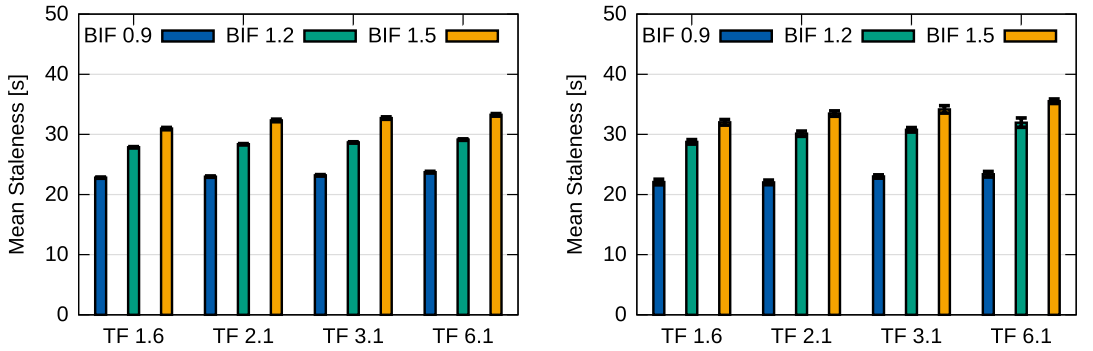
performance does not justify this increase. Instead, we configure the parameter with a value of 1.2 and 1.5, despite the fact that these configurations lead to a degraded performance of both approaches, as presented in Section A.2.4. However, as already indicated during that discussion, we examine the influence of a higher blocking interval factor when combined with a timeout factor above the default configuration of 1.6. During the following plots, we abbreviate the timeout factor as *TF* and the blocking interval factor as *BIF*.

System parameter	Configuration
Timeout factor	1.6, 2.1, 3.1, 6.1
Blocking interval factor	0.9, 1.2, 1.5

Table 50: Overview on the identified configurations for the combined evaluation of both system parameters, comprising four configurations for timeout factor and three for the blocking interval factor, which results in 12 experiments in total. The underlined values represent the default configurations of both system parameters.

ACCURACY AND TIMELINESS

Figure 91a and 91b depict the mean relative node count error for P- and C-BLOCK-TREE.KOM and reveal the interesting fact that both approaches accurately monitor



(a) Mean staleness of the provided monitoring results of P-BlockTree.KOM

(b) Mean staleness of the provided monitoring results of C-BlockTree.KOM

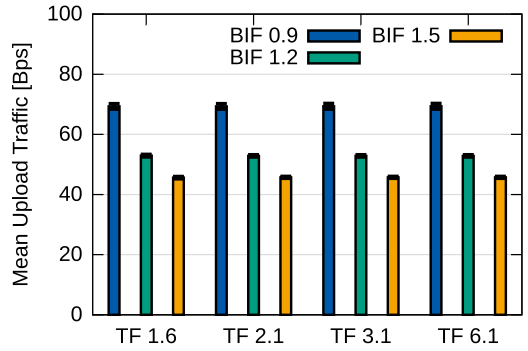
Figure 92: Overview on the impact of the combined parameter variation on the freshness of monitoring results

the number of nodes in the network even for larger blocking interval factors. However, the timeout factor must be chosen accordingly and should ideally be set to 3.1 or even 6.1. With a higher timeout factor the hierarchy and result tables avoid an early deletion of data so that the decelerated data collection and result dissemination do not lead to the previously observed data loss accompanied by the degrading performance. Instead, the influence of a delayed transmission of Aggregating-Up and Disseminating-Down messages mitigates with a higher timeout factor. In contrast to this outcome the results for the mean relative UDSF error (cf. Figure 91c and 91d) and for the mean staleness (cf. Figure 92a and 92b) unveil that a higher timeout factor does not compensate the delayed and decelerated data collection and result dissemination. Consequently, the mean relative UDSF error and the mean staleness increase for a higher blocking interval factor. A higher timeout factor amplifies this effect, because hierarchy and result tables may contain older information. However, the combined impact of both parameters on UDSF and staleness is rather small, as already indicated by Table 22 and 23 for P- and C-BlockTree.KOM, which list the proportion of variation of the system parameters from both approaches.

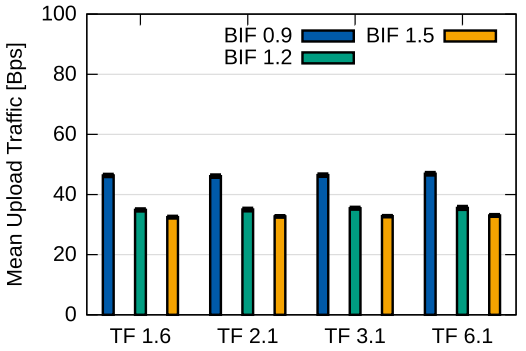
TRAFFIC

Figure 93 shows the well-known and already discussed results in terms of traffic if both parameters are separately examined. The timeout factor has no impact on P-BlockTree.KOM's traffic and slightly increases the traffic of C-BlockTree.KOM as a function of the factor. With a longer blocking interval factor the arising upload and download traffic of both approaches are reduced, because the parameter prevents an early and frequent transmission of messages.

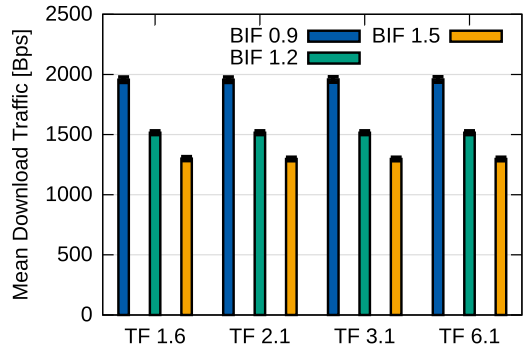
If both parameters are considered together and linked with the results for the performance metrics, two interesting findings are made. In terms of the mean relative UDSF error and staleness, we observe that both metrics increase as a function of the timeout factor for all configurations of the blocking interval factor. However, the positive influence of both parameters is observed based on the outcome of the mean relative node count error. Paired with a higher blocking interval factor the differences shrink for the varying configurations of the timeout factor in terms of the mean relative node count error. In terms of P-BlockTree.KOM, the mean relative



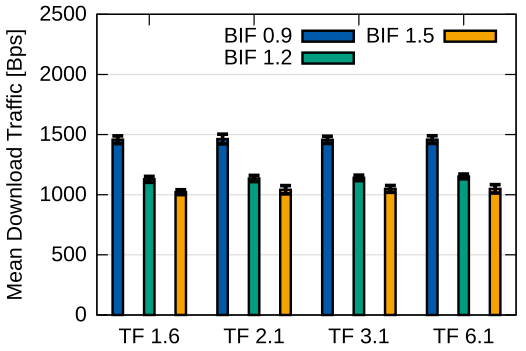
(a) Mean overall upload traffic per node of P-BlockTree.KOM



(b) Mean overall upload traffic per node of C-BlockTree.KOM



(c) Mean overall download traffic per node of P-BlockTree.KOM



(d) Mean overall download traffic per node of C-BlockTree.KOM

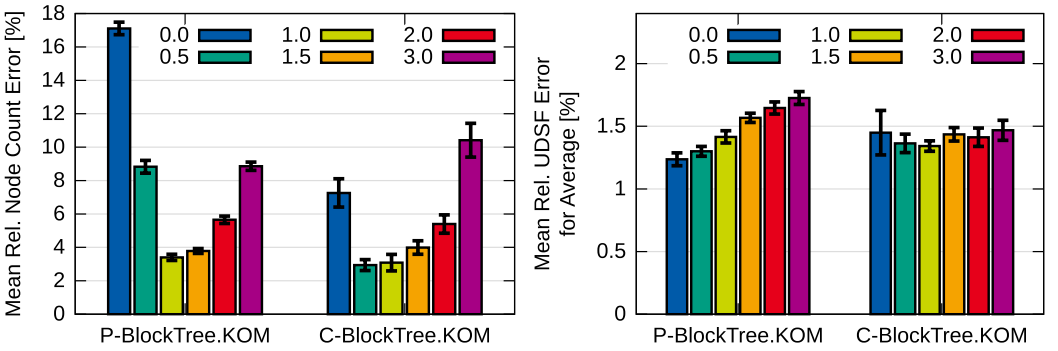
Figure 93: Overview on the overall mean traffic per node for the combined parameter variation

node count error for the combination TF = 6.1 and BIF = 1.2 nearly catches up with TF = 1.6 and BIF = 0.9, while the arising upload and download traffic for the first configuration are significantly reduced (cf. Figure 93a and 93c). In terms of C-BlockTree.KOM, the positive influence of both parameters becomes even more obvious. For TF = 6.1 the performance of a blocking interval factor of 1.2 or 1.6 is comparable to the performance for a factor 0.9 at considerably lower cost (cf. Figure 93b and 93d).

To summarize the observations in terms of performance and cost the combined variation of both parameters constitutes an effective alternative to reduce the arising traffic, while preserving the accuracy in terms of monitoring the number of nodes. However, the results additionally reveal that the observed efficiency comes at a higher mean staleness and an increased mean relative UDSF error, because both parameters cannot compensate the decelerated data collection and result dissemination.

A.2.6 *Operation Blocking Interval Factor*

The operation blocking interval factor represents the last system parameter of BlockTree.KOM that is evaluated. It represents a factor, which is multiplied with the aggregation interval parameter to define the effective interval for the operation blocking interval $t_{blockOperation}$. The operation blocking interval specifies how long a node must



(a) Mean relative node count error

(b) Mean relative UDSF error using the average as aggregation function

Figure 94: Overview on accuracy of the provided monitoring results for a varying operation blocking interval factor

wait before it is allowed to trigger an Aggregating-Up or Disseminating-Down operation after entering a new block. With this parameter BlockTree.KOM prevents that incomplete or wrong data are spread in the networks from nodes, which just entered a new block.

Table 51 lists the different configurations of the parameter with the default configuration of one, which corresponds to a complete aggregation interval before a node triggers an operation on its own. During the evaluation, we assess the impact of a shorter and no operation blocking interval on performance and cost by reducing the factor as well as the influence of a higher factor.

System parameter	Configuration
Operation blocking interval factor	0.0, 0.5, 1.0, 1.5, 2.0, 3.0

Table 51: System parameter variation of the operation blocking interval factor. The underlined value represents the default configuration of the system parameter.

Accuracy and Staleness

Based on the results in Figure 94a, which show the mean relative node count error, it becomes apparent that the influence of this parameter differs to the influence of the previously discussed parameters. Contrary to these parameters, where the variation has an unidirectional effect on the mean relative node count error, the results indicate that the best performance is obtained by choosing a value from the middle of the configuration range. In terms of no or short intervals, a node, which entered a new block, has insufficient information that do not reflect the state of its block or sector. As a result triggering an Aggregating-Up or Disseminating-Down operation based on this current state leads to the collection and dissemination of incomplete data. In contrast, longer operation blocking intervals prevent a node from participating in collecting data and disseminating results for a longer amount of time. Due to the long interval, it might happen that multiple nodes inside a block are currently not allowed to trigger operations. In the worst case the monitoring information from that block cannot be collected or disseminated and gets lost, though, the block is populated with nodes. Figure 94a also reveals that operation blocking intervals be-

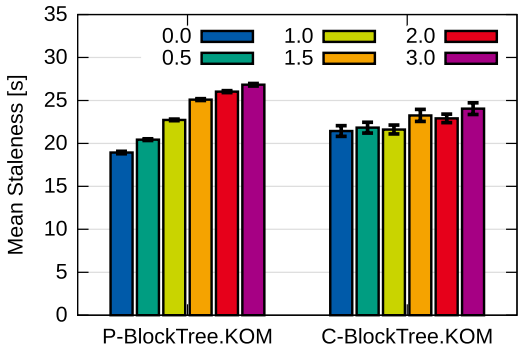


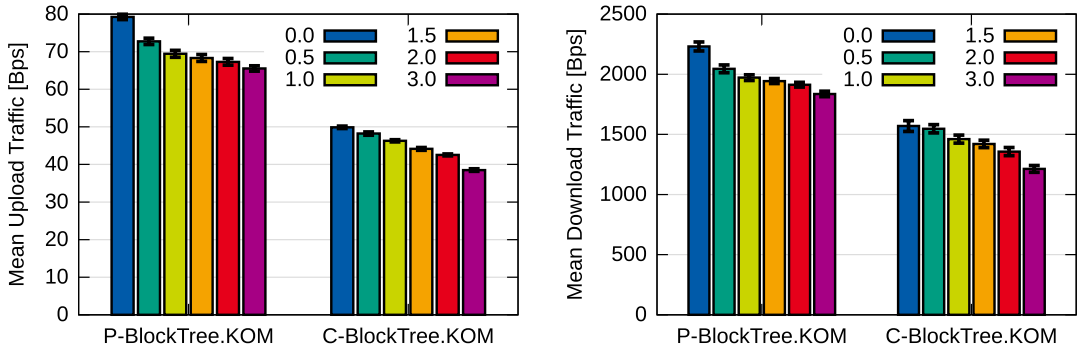
Figure 95: Mean staleness of the provided monitoring results for a varying operation blocking interval factor

low the default configuration of the parameter have a stronger impact on P- than on C-BlockTree.KOM. The stronger influence on P-BlockTree.KOM arises from the periodic tasks per block, every node has to execute. In C-BlockTree.KOM every block just triggers one Aggregating-Up operation to transmit its data to the corresponding concentrating block and only the highest concentrating block is in charge to disseminate the results. In contrast, every block of P-BlockTree.KOM participates at every active level of the hierarchy. Consequently, a node with insufficient information triggers an Aggregating-Up operation at every level, which leads to the collection of incomplete data at the remaining sectors per level and becomes apparent by the increased mean relative node count error.

The results for the mean relative UDSF error (cf. Figure 94b) outline again that missing values and incomplete data do not severely influence the global view of this attribute. It suffices to collect only a fraction of data to reflect the effective global view of that attribute. With a shorter blocking interval in terms of P-BlockTree.KOM the waiting time for nodes is reduced so that they contribute earlier with their local information, which becomes apparent by the smaller mean relative error as well as by the smaller mean staleness of the results (cf. Figure 95). With a longer operation blocking interval both the mean relative UDSF error and the mean staleness increase. The results of UDSF and staleness reflect again the strong interdependence between both metrics, which indicates that a faster collection and dissemination improves at least the accuracy of this attribute. The same interdependence is also observed for C-BlockTree.KOM, however, this approach does not benefit that much from shorter operation blocking intervals, because the results do not reflect the gain in accuracy and timeliness, as for P-BlockTree.KOM. As stated above, the smaller influence arises from the periodic tasks per block in C-BlockTree.KOM, because a node with incomplete information is only a member of one level and not of all active levels. Thus, the influence of collecting incomplete information is limited to one and not to all active levels.

TRAFFIC

The results for the mean upload and download traffic in Figure 96a and 96b also reveal for this parameter that delaying the participation from a fraction of nodes reduces the arising upload and download traffic. The illustrated reduction is primarily attributed to the reduced traffic for the Aggregating-Up operations, while the impact on the Disseminating-Down traffic in terms of C-BlockTree.KOM is negli-



(a) Mean overall upload traffic per node

(b) Mean overall download traffic per node

Figure 96: Overview on the overall traffic per node for a varying operation blocking interval factor

gible. In general, the impact of this parameter on the arising traffic is not that strong compared to the other parameters, such as the aggregation interval and the blocking interval factor, which configure the information exchange between the nodes and considerably influence the overall upload and download traffic. The operation blocking interval factor just influences a fraction of nodes, whereas the two mentioned parameters affect the behavior of all nodes.

A.3 MOBI-G.KOM: EVALUATION OF THE DIFFERENT TRAFFIC TYPES

Contrary to BLOCKTREE.KOM, this section does not deal with the evaluation of further system parameters, which have not been examined in the evaluation chapter. Instead, we just focus on MOBI-G.KOM's different traffic types. For the examination and ranking of the different types of traffic we evaluate the results from the scenarios with and without churn. Table 52 and 53 depict the overall upload and download traffic, which are partitioned into the four listed traffic types, where the corresponding figures represent the mean upload and download traffic per second per node. Contrary to the two approaches of BLOCKTREE.KOM, the types of traffic are the same for MOBI-G.KOM DM and CM.

	Mobi-G.KOM DM		**Mobi-G.KOM CM**	
Traffic type [Bps]	**No churn**	**Churn**	**No churn**	**Churn**
Overall traffic	22.57	24.22	50.91	49.74
IC traffic	1.16	1.84	10.88	11.17
Ack traffic	0.37	0.47	3.63	3.55
IS traffic	12.16	13.14	26.11	24.90
Army traffic	8.73	8.54	8.72	8.52

Table 52: Classification of the overall arising upload traffic into IC, acknowledgment (ack), IS, and army traffic

	Mobi-G.KOM DM		**Mobi-G.KOM CM**	
Traffic type [Bps]	**No churn**	**Churn**	**No churn**	**Churn**
Overall traffic	464.08	505.81	766.88	790.12
IC traffic	1.05	1.37	10.52	10.38
Ack traffic	0.36	0.58	3.54	3.64
IS traffic	270.30	306.99	560.72	588.82
Army traffic	192.37	196.88	192.11	196.23

Table 53: Classification of the overall arising download traffic into IC, acknowledgment (ack), IS, and army traffic

Based on the classification, it becomes apparent for both variants that the transmission of IC messages and their corresponding acknowledgments (acks) just account for a small amount of the traffic, whereas the largest amount results from the dissemination of IS messages followed by the army messages. The difference becomes particularly apparent for the download traffic, because IS and army messages are broadcasted and received by multiple nodes. In contrast, the gap between the amount of sent and received IC and ack messages is considerably smaller, because these messages are always exchanged between two nodes, using unicast. Looking at the differences between MOBI-G.KOM DM and CM, the results uncover that the overall larger amount of traffic is attributed to the higher amount of IC, ack, and IS

messages. Again, the IS traffic accounts for the largest part of the traffic, however, the results outline the influence on the IC and ack traffic as well, because the message exchange is not limited to the beginning of an epoch, as for discrete monitoring, but takes place during the whole epoch. In contrast to this observation the outcome for the army traffic unveils that the identification of a beacon and its corresponding army is neither influenced by discrete nor continuous monitoring. As expected and reflected by the results the identification of the beacon and its army does not depend on the type of monitoring and if tokens are exchanged only at the beginning or during the whole epoch.

B

AUTHOR'S PUBLICATIONS

B.1 MAIN PUBLICATIONS

1. Dominik Stingl, Christian Gross, Leonhard Nobach, Ralf Steinmetz, and David Hausheer. BlockTree: Location-Aware Decentralized Monitoring in Mobile Ad Hoc Networks. In *Proceedings of the 38th IEEE Conference on Local Computer Networks*, 2013.

2. Dominik Stingl, Christian Gross, Julius Rückert, Leonhard Nobach, Aleksandra Kovacevic, and Ralf Steinmetz. PeerfactSim.KOM: a Simulation Framework for Peer-to-Peer Systems. In *Proceedings of the International Conference on High Performance Computing and Simulation*, 2011.

3. Dominik Stingl, Christian Gross, and Karsten Saller. Decentralized Monitoring in Peer-to-Peer Systems. In Wolfgang Effelsberg, Ralf Steinmetz, and Thorsten Strufe, editors, *Benchmarking Peer-to-Peer Systems*, pages 81–113. Springer, 2013.

4. Dominik Stingl, Christian Gross, Karsten Saller, Sebastian Kaune, and Ralf Steinmetz. Benchmarking Decentralized Monitoring Mechanisms in Peer-to-Peer Systems. In *Proceedings of the Joint WOSP/SIPEW International Conference on Performance Engineering*, 2012.

5. Dominik Stingl, Reimond Retz, Björn Richerzhagen, Christian Gross, and Ralf Steinmetz. Mobi-G: Gossip-Based Monitoring in MANETs. In *IEEE/IFIP Network Operations and Management Symposium*, 2014.

6. Dominik Stingl, Björn Richerzhagen, Fabio Zöllner, Christian Gross, and Ralf Steinmetz. PeerfactSim.KOM: Take it Back to the Streets. In *Proceedings of the International Conference on High Performance Computing and Simulation*, 2013.

B.2 CO-AUTHORED PUBLICATIONS

7. Osama Abboud, Konstantin Pussep, Dominik Stingl, and Ralf Steinmetz. Media-aware Networking for SVC-Based P2P Streaming. In *Network and Operating System Support for Digital Audio and Video*, 2011.

8. Kalman Graffi, Christian Gross, Dominik Stingl, Daniel Hartung, Aleksandra Kovacevic, and Ralf Steinmetz. LifeSocial.KOM: a Secure and P2P-Based Solution for Online Social Networks. In *Proceedings of the IEEE Consumer Communications and Networking Conference*, 2011.

9. Kalman Graffi, Dominik Stingl, Christian Gross, Hoang Nguyen, Aleksandra Kovacevic, and Ralf Steinmetz. Towards a P2P Cloud: Reliable Resource Reservations in Unreliable P2P Systems. In *Proceedings of the 16th IEEE International Conference on Parallel and Distributed Systems*, 2010.

10. Kalman Graffi, Dominik Stingl, Julius Rückert, Aleksandra Kovacevic, and Ralf Steinmetz. Monitoring and Management of Structured Peer-to-Peer Systems. In *Proceedings of the 9th International Conference on Peer-to-Peer Computing*, 2009.

11. Christian Gross, Fabian Kaup, Dominik Stingl, Björn Richerzhagen, David Hausheer, and Ralf Steinmetz. EnerSim: an Energy Consumption Model for Large-Scale Overlay Simulators. In *Proceedings of the 38th IEEE Conference on Local Computer Networks*, 2013.

12. Christian Gross, Max Lehn, Dominik Stingl, Aleksandra Kovacevic, Alejandro P. Buchmann, and Ralf Steinmetz. Towards a Common Interface for Overlay Network Simulators. In *Proceedings of the 16th IEEE International Conference on Parallel and Distributed Systems*, 2010.

13. Christian Gross, Björn Richerzhagen, Dominik Stingl, Christoph Münker, David Hausheer, and Ralf Steinmetz. Geodemlia: Persistent Storage and Reliable Search for Peer-to-Peer Location-Based Services. In *Proceedings of the 13th IEEE Conference on Peer-to-Peer Computing*, 2013.

14. Christian Gross, Björn Richerzhagen, Dominik Stingl, Jan Weber, David Hausheer, and Ralf Steinmetz. GeoSwarm: a Multi-Source Download Scheme for Peer-to-Peer Location-Based Services. In *Proceedings of the 13th IEEE Conference on Peer-to-Peer Computing*, 2013.

15. Christian Gross, Dominik Stingl, Wolfgang Effelsberg, and Ralf Steinmetz. Neuer DFG-Sonderforschungsbereich an der Technischen Universität Darmstadt. *Praxis der Informationsverarbeitung und Kommunikation.*, 36(1):65–66, 2013.

16. Christian Gross, Dominik Stingl, Christian Gottron, Björn Richerzhagen, Christoph Münker, and David Hausheer. Harnessing Mobile Ad Hoc Communication for Decentralized Location-Based Services. Technical report, Peer-to-Peer Systems Engineering, Nov 2012.

17. Christian Gross, Dominik Stingl, Björn Richerzhagen, Andreas Hemel, Ralf Steinmetz, and David Hausheer. Geodemlia: a Robust Peer-to-Peer Overlay Supporting Location-Based Search. In *Proceedings of the 12th IEEE International Conference on Peer-to-Peer Computing*, 2012.

18. Max Lehn, Tonio Triebel, Christian Gross, Dominik Stingl, Karsten Saller, Wolfgang Effelsberg, Aleksandra Kovacevic, and Ralf Steinmetz. Designing Benchmarks for P2P Systems. In Kai Sachs, Ilia Petrov, and Pablo Guerrero, editors, *From Active Data Management to Event-Based Systems and More*, pages 209–229. Springer, 2010.

19. Karsten Saller, Dominik Stingl, and Andy Schürr. D4M, a Self-Adapting Decentralized Derived Data Collection and Monitoring Framework. In *Workshops der wissenschaftlichen Konferenz Kommunikation in Verteilten Systemen*, 2011.

20. Matthias Wichtlhuber, Julius Rückert, Dominik Stingl, Matthias Schulz, and David Hausheer. Energy-Efficient Mobile P2P Video Streaming. In *Proceedings of the 12th International Conference on Peer-to-Peer Computing*, 2012.

C

CURRICULUM VITÆ

PERSONAL INFORMATION

Name	Dominik Stingl
Date of Birth	October 5, 1981
Place of Birth	Mainz
Nationality	German

EDUCATION

since 02/2013	Technische Universität Darmstadt, Germany Doctoral candidate at the department of Electrical Engineering and Information Technology
09/2013 – 11/2013	University of Massachusetts Amherst, USA Visiting researcher at the department of Electrical and Computer Engineering
10/2002 – 11/2008	Technische Universität Darmstadt, Germany Studies of computer science Degree: Diplom-Informatiker (Dipl.-Inform.)
08/1992 – 06/2001	Gutenberg-Gymnasium, Wiesbaden, Germany Degree: Abitur, majoring in mathematics and French

PROFESSIONAL EXPERIENCE

since 01/2013	Technische Universität Darmstadt, Germany Managing Director of the DFG Collaborative Research Center 1053 "MAKI – Multi-Mechanisms Adaptation of the Future Internet" at the Multimedia Communications Lab together with Christian Groß
since 10/2012	Technische Universität Darmstadt, Germany Head of the research group "Adaptive Overlay Communications" at the Multimedia Communications Lab together with Christian Groß
since 05/2009	Technische Universität Darmstadt, Germany Research assistant at the Multimedia Communications Lab
02/2009 – 05/2009	Technische Universität Darmstadt, Germany Student assistant at the Multimedia Communications Lab
11/2007 – 03/2008	NEC Laboratories Europe, Heidelberg, Germany Internship

07/2001 – 04/2002	Asklepios Paulinen Hospital, Wiesbaden, Germany Civilian service

AWARDS AND HONORS

11/2009	Best diploma thesis award from the "Darmstädter Stiftung für Technologietransfer" for the diploma thesis "Development of a Self-Optimizing Life Cycle Framework for Structured Peer-to-Peer Systems"

SCIENTIFIC ACTIVITIES

Organization	Technical Program for the Information Technology Society (ITG) at the VDE Congress, 2010
PC Member	Workshop on Modeling and Simulation of Peer-to-Peer and Autonomic Systems (MOSPAS), 2014
Reviewer	IEEE/IFIP Networking Operations and Management Symposium (NOMS), 2012
	Journal "Future Generation Computer Systems", 2012
	IEEE International Conference on Peer-to-Peer Computing (P2P), 2011 – 2013
	ACM Multimedia Systems Conference (MMSys), 2013
	IEEE Symposium on Computers and Communications (ISCC), 2014

TEACHING ACTIVITIES

10/2012 – 03/2013	Technische Universität Darmstadt, Germany Tutor for the lab "Multimedia Communications Lab"
04/2012 – 09/2012	Technische Universität Darmstadt, Germany Teaching assistant for the lecture "Introduction to Net Centric Systems"
10/2011 – 03/2012	Technische Universität Darmstadt, Germany Tutor for the lab "SmartNets Lab"
04/2011 – 09/2011	Technische Universität Darmstadt, Germany Tutor for the seminar "Selected Topics in P2P Research"
04/2010 – 09/2010	Technische Universität Darmstadt, Germany Tutor for the lab "Multimedia Communications Lab"
04/2010 – 09/2010	Technische Universität Darmstadt, Germany Tutor for the seminar "Benchmarking P2P"
10/2009 – 03/2010	Technische Universität Darmstadt, Germany Tutor for the lab "Multimedia Communications Lab"

SUPERVISED THESES

KOM-D-0409 Tobias Weber, *Development and Execution of a Performance Analysis for Server-Based Mobile Payment Architectures*. Diploma Thesis, Technische Universität Darmstadt, November 2010.

KOM-D-0442 Johann Degenhardt, *Energy-Aware Location-Based Routing and Data Management in Peer-to-Peer Systems*. Diploma Thesis, Technische Universität Darmstadt, January 2012.

KOM-D-0447 Leonhard Nobach, *Location-Aware Data Aggregation in Mobile P2P Networks*. Master Thesis, Technische Universität Darmstadt, April 2012.

KOM-D-0466 Fabio Zöllner, *Location-Centric Replication in Mobile Peer-to-Peer Networks*. Master Thesis, Technische Universität Darmstadt, May 2013.

KOM-M-0500 Nils Richerzhagen, *Adaptive Monitoring for Mobile Networks in Challenged Environments*. Master Thesis, Technische Universität Darmstadt, November 2014.

KOM-S-0385 Muain Alhaddad, *Survey and Classification of Decentralized Monitoring Approaches*. Bachelor Thesis, Technische Universität Darmstadt, April 2011.

KOM-S-0407 Dieter Hofmann, *Gossip-Based Monitoring in Mobile Peer-to-Peer Networks*. Bachelor Thesis, Technische Universität Darmstadt, August 2011.

KOM-S-0415 Alexander Nigl, *Probabilistic Node Counting in P2P Systems*. Bachelor Thesis, Technische Universität Darmstadt, November 2011.

KOM-S-0417 Matthias Krumrein, *Design, Realization, and Analysis of a Time Synchronization Approach for a Hybrid P2P Simulation Environment*. Bachelor Thesis, Technische Universität Darmstadt, November 2011.

KOM-S-0418 Andreas Schneider, *Design and Implementation of a Communication Interface Between a Prototypical P2P Implementation and the Corresponding P2P Simulation Model*. Bachelor Thesis, Technische Universität Darmstadt, November 2011.

KOM-S-0454 Reimond Retz, *Gossip-Based Monitoring: Design and Development of an Extension for Mobile Peer-to-Peer Networks*. Bachelor Thesis, Technische Universität Darmstadt, February 2013.

Darmstadt, July 18, 2014

Dominik Stingl

D

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

Ich versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

Darmstadt, 18. Juli 2014

Dominik Stingl